UNIVERSITÀ DI PISA

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea Specialistica in Informatica

Tesi di Laurea

Simulazione combinatoria a vincoli di mappe di interazione molecolare

Relatore:
Prof. PIERPAOLO DEGANO

Laureando:
SIMONE FONDA

Correlatori:
Prof. ALBERTO POLICRITI
Dott. LUCA BORTOLUSSI

ANNO ACCADEMICO 2006-2007

Dedicato,
ancora una volta,
alla mia splendida Famiglia.

Indice

Elenco delle figure

Elenco delle tabelle

Elenco dei listati

Introduzione

Nella moderna *System Biology* c'è la necessità di adottare una notazione che diventi lo standard per la rappresentazione delle reti di regolazione biologica. Questa notazione dovrebbe permettere ai biologi di esprimere con semplicità i molteplici meccanismi che regolano il funzionamento della cellula, in modo da poter condividere e diffondere con successo l'incalzante susseguirsi di scoperte di cui questo ramo della scienza sta attualmente godendo. Questo standard, inoltre, dovrebbe fungere da punto di incontro tra biologi e bioinformatici, vista l'importanza crescente che gli esperimenti *in silico*, cioè al computer, stanno acquistando al giorno d'oggi.

Nell'ampio panorama delle proposte si stagliano le Mappe di Interazione Molecolare (MIM), sviluppate da Kurt W. Kohn. Le mappe non sono null'altro che dei grafi in cui i nodi rappresentano le entità in gioco, come proteine, filamenti di dna o molecole in generale. Gli archi definiscono come queste entità interagiscono tra loro, descrivendo legami covalenti e non covalenti, modificazioni come la fosforilazione o la conversione stechiometrica ed altro ancora.

I metodi maggiormente utilizzati per simulare al calcolatore questa ed altre notazioni simili si possono suddividere in due principali famiglie. La prima prevede l'utilizzo delle Equazioni Differenziali Ordinarie (ODE). Chi le utilizza deve necessariamente specificare minuziosamente tutti i dettagli di ogni reazione, per poter definire un'equazione, compito che spesso ruba più tempo della simulazione stessa. Inoltre le ODE trattano necessariamente come continue entità che altrimenti verrebbero considerate discrete, come per esempio il numero di copie di una certa proteina.

La seconda famiglia è rappresentata dalle simulazioni stocastiche, che si adattano perfettamente alle entità biologiche, notoriamente non deterministiche. In genere sono computazionalmente più costosi ma più precisi ed aderenti alla realtà che si sta modellando.

Nella presente tesi viene proposta una codifica delle Mappe di Interazione Molecolare scritta in un linguaggio stocastico, concorrente ed a vincoli: sCCP. Le mappe sono compatte: i simboli implicitamente descrivono un gran numero di reazioni e non sono semplici da gestire con gli strumenti standard. La codifica proposta gestisce la simulazione in modo stocastico mantenendo implicita la lista delle possibili reazioni, quindi mantenendo la dimensione della codifica stessa proporzionale alla dimensione della mappa e non all'insieme di reazioni che implicitamente descrive. Questo ultimo aspetto è cruciale: come si vedrà le mappe hanno una semantica complessa e descrivono un gran numero di reazioni tramite pochi simboli, generando una esplosione combinatoria.

Dopo aver introdotto la notazione ed aver visto qualche esempio, verranno discusse le interpretazioni che è possibile dare ad ogni mappa: una esplicita che costringe a disegnare molti più simboli ma è più semplice da trattare, ed una combinatoria che implicitamente rappresenta con pochi nodi ed archi intere famiglie di reazioni diverse.

Si adotterà quest'ultima interpretazione e la codifica che si fornirà cercherà di mantenere questa informazione nascosta, senza esplicitarla, al fine di mantenere bassa la complessità del codice sorgente necessario ad effettuare una simulazione. Si cercherà quindi di sfruttare questo aspetto implicito e combinatorio delle mappe all'interno della codifica stessa.

Verrà introdotto sCCP, una estensione stocastica di CCP, la nota algebra di processo. Verrà brevemente introdotta la sintassi e le catene di Markov alla base della sua semantica.

In seguito si presenterà la codifica proposta, vedendo come si comporta su alcuni semplici esempi ed analizzando nel dettaglio il funzionamento dei predicati e degli agenti che la costituiscono.

Verrà poi presentato un esempio concreto di simulazione, esibendo i dettagli relativi alla definizione dell'ambiente di simulazione ed alla sua esecuzione, oltre che alla raccolta dei risultati.

Infine si discuteranno alcuni dei metodi simili per approccio o per tecniche utilizzate, attualmente esistenti per la simulazione di entità biologiche.

Capitolo 1

Mappe di interazione molecolare (MIM)

Le Mappe di Interazione Molecolare (Molecular Interaction Maps, MIM, o mappe di Kohn [18]) sono nate con l'obiettivo di rappresentare, tramite dei semplici diagrammi, reti di regolazione biologica. La semplicità e la flessibilità della sintassi le rendono uno dei più potenti strumenti di questo tipo disponibili ad oggi, al costo di una semantica non sempre matematicamente rigorosa. Fortunatamente questo problema può venir risolto semplicemente assumendo alcune ipotesi sui sistemi analizzati, come verrà descritto in seguito.

1.1 L'ambito di utilizzo delle MIM

Quotidianamente vengono scoperti talmente tanti nuovi dettagli sul comportamento delle migliaia di molecole studiate, che risulta difficile organizzare tale conoscenza in modo da poterla sfruttare al meglio. Le mappe si collocano proprio in questo ambito: rappresentare in modo conciso ma efficace le interazioni tra molecole. Volendo fare un parallelo, si potrebbe dire che le MIM si prefiggono di diventare per questo ramo della biologia quello che per l'elettronica sono i diagrammi utilizzati per descrivere i circuiti elettronici. Una notazione con queste caratteristiche consentirebbe una accurata e precisa rappresentazione dell'informazione legata a sistemi biologici, studiati a livello molecolare, velocizzando il processo di stesura di modelli biologici e della loro verifica sperimentale, anche grazie alla simulazione automatizzata tramite computer.

Il compito non è facile: le reti di regolazione molecolare esprimono i dettagli dei meccanismi con cui si formano complessi molecolari e come questi ultimi interagiscono tra loro. Descrivono inoltre i possibili modi in cui vengono mo-

dificate le proteine, per esempio tramite fosforilazione, i dettagli legati ai loro domini proteici ed altro ancora.

Questo ramo della ricerca è relativamente giovane e non si è ancora imposto uno standard universalmente riconosciuto. I formalismi con obiettivi più o meno simili sono parecchi ([11], [1], [7]) ognuno con le proprie peculiarità che li rende più o meno adatti a seconda dei casi. I parametri con cui viene scelta una notazione sono diversi: rappresentazione discreta o continua delle entità, possibilità di simulazione e visualizzazione al computer, disponibilità di rigorose tecniche matematiche di verifica, livello di astrazione e di dettaglio possibili e così via. Alcune caratteristiche, come la composizionalità, hanno dimostrato la propria efficacia nel descrivere sistemi biologici, soprattutto a livello molecolare, e sono quindi da ricercare. Infine non bisogna sottovalutare la pesantezza della notazione misurata proprio in termini di numero di simboli: un formalismo eccessivamente prolisso potrebbe avere il pregio di riuscire a descrivere ogni minimo dettaglio, al costo di una difficoltà enorme nella stesura dei modelli stessi.

Per esempio, poche proteine si possono combinare tra loro in molti modi diversi, generando un elevatissimo numero di complessi, ognuno dei quali interagisce in modo diverso con ciò che gli sta intorno. Riuscire a districarsi nell'ingrediente combinatorio presente in questi problemi, al fine di elencare ogni possibile interazione, potrebbe essere addirittura computazionalmente impossibile.

Un aspetto chiave è quindi la simulazione al computer. Oggi giorno, vista la mole di informazioni da trattare, sta diventando sempre più importante avere a disposizione degli strumenti da poter affiancare (ed in alcuni casi addirittura rimpiazzare) all'analisi di laboratorio, gli esperimenti *in vitro*. Sebbene i formalismi a disposizione siano tra loro anche profondamente diversi, la semantica sottostante è molto spesso la stessa con qualche piccola variazione.

Principalmente le possibilità sono due. La prima è rappresentata senza dubbio dalle *Equazioni Differenziali Ordinarie* (ODE) ([11], [30], [19]), grazie alle ormai rodatissime ed efficienti tecniche di analisi e soluzione, si è dimostrata in questi ultimi anni un potente mezzo di indagine di sistemi biologici. Purtroppo quello che manca a questo approccio è la flessibilità: per definire in modo rigoroso il sistema è necessario un lungo processo di stesura delle equazioni che, anche per sistemi semplici, si può rivelare decisamente pesante. L'altra grande famiglia di metodi si basa su processi stocastici, appoggiandosi di solito su *catene di Markov a tempo continuo* ([22]). Questo approccio è considera-

to più realistico vista la natura dei sistemi biologici analizzati e sfrutta una variante di un semplice quanto efficace algoritmo di simulazione: l'algoritmo di Gillespie ([12], [13]), già utilizzato da tempo per maneggiare con successo reazioni chimiche. A differenza delle ODE, i processi stocastici descrivono le entità come discrete e seguono una evoluzione non predeterminata: hanno appunto una natura stocastica.

Le Mappe di Interazione Molecolare offrono la possibilità di sfruttare entrambi questi approcci. La simulazione tramite ODE è stata formalizzata già da Kohn ([21]) ed utilizzata per diversi casi di studio ([19], [20]). La possibilità di sfruttare processi stocastici verrà invece approfondita nella presente tesi.

Nel farlo verranno associati dei *rate* ad ogni simbolo di reazione. Tali rate sono dei valori che esprimono in un certo la velocità con la quale una certa reazione accadrà, oppure possono venire interpretati come il tempo che bisogna aspettare prima che accada. Una seconda interpretazione, più affine alla semantica del linguaggio utilizzato per codificare le mappe, prevede che vengano considerati come delle priorità. In pratica il sistema modellato è visto come un insieme di reazioni ognuna con un rate associato, che competono in modo stocastico per avvenire. Questa gara verrà vinta la maggior parte delle volte (in senso stocastico) da chi ha il rate più alto, per cui, in un certo senso, da chi ha la priorità maggiore.

Questa particolare proprietà è una delle più difficili da indagare in un sistema biologico, richiede spesso esperimenti specifici che non sempre vengono eseguiti, preferendo utilizzare stime. Il nostro approccio prevede che i *rate* siano conosciuti a priori e vengano associati da chi modella il sistema ad ogni reazione. Come si vedrà, essi sono una parte fondamentale della buona riuscita della simulazione.

1.2 La sintassi

Qui di seguito viene presentata la notazione elaborata da Kohn. Tuttavia, per una trattazione completa ed esaustiva, si rimanda a [18].

1.2.1 Specie molecolari e interazioni

Le mappe, come detto, sono dei semplici grafi, in cui i nodi rappresentano le molecole e gli archi le interazioni. Ogni tipo di molecola appare generalmente una volta sola all'interno della mappa, permettendo di concentrare tutte le

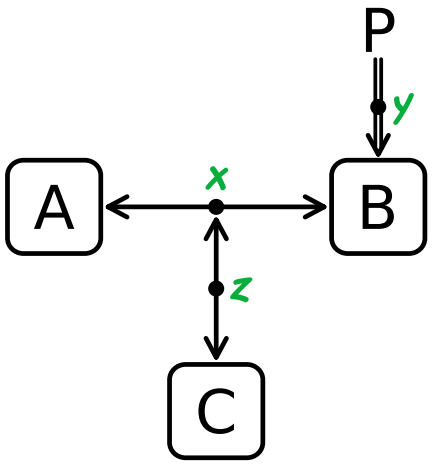


Figura 1.1: Una semplice Mappa di Interazione Molecolare

interazioni che la coinvolgono in un unico punto.

Una *specie molecolare elementare* è dunque descritta tramite un nodo; in Figura 1.1 i nodi A, B e C sono tre diverse specie molecolari elementari.

Diversi tipi di interazione tra molecole sono indicate tramite diversi tipi di archi i quali si possono distinguere grazie ai simboli posti agli estremi. Il punto di contatto tra frecce e nodi rappresenta un sito di interazione; anche per essi vale la proprietà di unicità: ogni sito di interazione viene rappresentato una volta sola all'interno di ogni molecola. Qualche semplice esempio di interazione: la doppia freccia uncinata (Figura 1.2a) indica la reazione di *complessazione*[1] tra due molecole mentre una freccia singola con una P all'altra estremità (Figura 1.2b) indica una generica *fosforilazione*[2]. Gli archi di interazione si possono incrociare ma il loro significato non viene modificato, in pratica è come se non lo facessero.

Per indicare il prodotto di una interazione si pongono uno o più nodi sull'arco stesso (porne più d'uno può essere utile per mantenere la mappa ordinata). Per esempio in Figura 1.1 il nodo x rappresenta la specie molecolare elementare A legata tramite una reazione di complessazione alla specie molecolare

[1]Con il termine complessazione verrà indicato, per brevità, il legame non covalente e reversibile tra molecole.

[2]La fosforilazione è una reazione chimica che consiste nell'addizione di un gruppo fosfato ad una proteina o ad un'altra molecola, in pratica viene aggiunto un elemento alla molecola originale tramite un legame covalente.

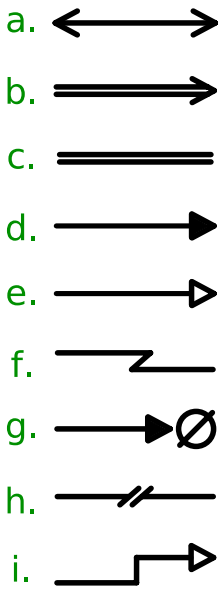


Figura 1.2: Simboli di reazione: a. legame non covalente; b. fosforilazione; c. legame covalente; d. conversione stechiometrica; e. produzione senza perdita di reagenti; f. divisione di un legame covalente; g. degradazione; h. reazione in trans (intermolecolare); i. trascrizione.

elementare B. Il risultato è definito come *specie molecolare complessa* o, per non creare troppa confusione, *specie molecolare non elementare* e verrà indicata con la notazione $A : B$. Nella stessa figura, il nodo y è il risultato della reazione di fosforilazione della specie molecolare elementare B; tale prodotto verrà indicato, dove non ambiguo, con pB.

Definizione 1.2.1 *Una specie molecolare elementare è un tipo di molecola, indicato nelle Mappe di Interazione Molecolare come un nodo con un nome al suo interno.*

Definizione 1.2.2 *Una interazione è una reazione tra specie molecolari (elementari o meno) oppure una loro modificazione.*

Definizione 1.2.3 *Una specie molecolare non elementare è il prodotto di una interazione.*

Ogni nodo può essere oggetto di interazione. Tornando alla Figura 1.1 il nodo x, cioè il complesso $A : B$, è legato a C da una reazione di complessazione, dunque il nodo z rappresenta il complesso $A : B + C = A : B : C$. In questo modo è possibile modellare, per esempio, il caso in cui il legame tra

due proteine A e B modifica la forma di una delle due, scoprendo un sito di interazione a cui se ne può legare una terza, C. Si noti che la mappa dice chiaramente che C non si può legare ne ad A ne a B, ma solo ed esclusivamente al complesso $A : B$, una volta formato.

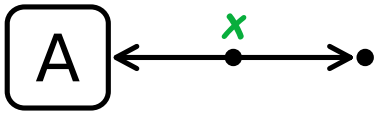


Figura 1.3: Il nodo isolato è una copia di A.

Per indicare un *omodimero*[3] senza rappresentare due volte la stessa molecola, viene utilizzato un nodo isolato legato da una interazione. In Figura 1.3 il nodo x rappresenta il complesso $A : A$, in cui il nodo isolato è in pratica una copia di A.

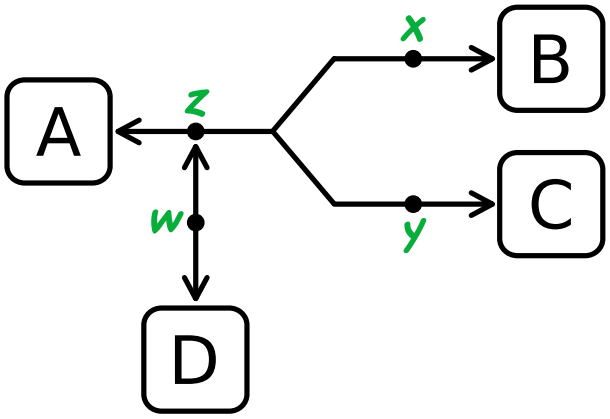


Figura 1.4: B e C competono per potersi legare ad A.

In Figura 1.4 viene espressa una mutua esclusione tra due reazioni cioè due molecole che competono per lo stesso sito di interazione. Il sito in questione appartiene alla molecola A e le due reazioni sono le complessazioni con B e C. La sintassi prevede l'unione delle due reazioni ad angolo acuto, il quale

[3]Un dimero è una molecola formata dalla generica unione di due subunità dette monomeri. Nel caso in cui esse siano di identica natura chimica il prodotto viene chiamato omodimero, quando la natura chimica è differente prende il nome di eterodimero.

indica che la mappa non esprime la reazione tra B e C. I nodi presenti sulle interazioni possono acquistare significati multipli. Mentre x e y indicano rispettivamente $A : B$ e $A : C$ il nodo w indica uno qualsiasi tra $A : B$ e $A : C$, in pratica uno dei possibili complessi formati sfruttando quel sito di interazione.

Questa notazione permette la rappresentazione in maniera molto compatta di legami esclusivi aventi la stessa funzione. Il nodo w infatti rappresenta i due trimeri $A : B : D$ e $A : C : D$: diversi complessi espressi tramite un unico simbolo.

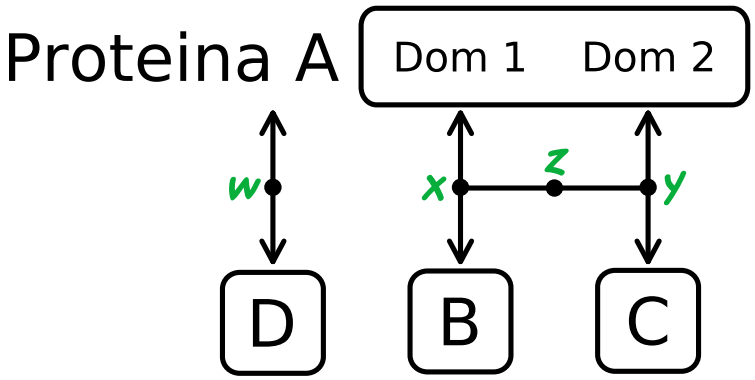


Figura 1.5: Legami specifici per dominio.

Spesso le proteine sono composte da domini con funzioni indipendenti. Per esplicitare questo tipo di informazioni è possibile usare la notazione di Figura 1.5: la proteina A è composta da due domini e si può legare alla molecola D in un sito sconosciuto (il prodotto $A : D$ è indicato con il nodo w), con B in un sito del dominio 1 ed infine, grazie ad un sito di interazione nel dominio 2, con C. Nella stessa figura è introdotto il simbolo di combinazione di stati: la linea semplice tra i due nodi x e y rappresenta la situazione in cui esistono sia x che y: nell'esempio è il caso in cui A è legata sia a B che a C. Grazie al nodo z si può quindi accedere al complesso $A : B : C$.

Le mappe possono esprimere anche legami tra domini di una stessa molecola. La Figura 1.6 mostra i due casi possibili: il primo è un legame *in cis*, cioè *intra*molecolare, all'interno della stessa molecola, mentre l'altro è un legame *in trans*, cioè *inter*molecolare e quindi tra due copie di una stessa molecola. Il nodo x di Figura 1.6 rappresenta dunque una molecola A in cui due suoi domini sono legati; y rappresenta l'omodimero formato tramite il legame del dominio 1 di una copia di B con il dominio 2 di una seconda B.

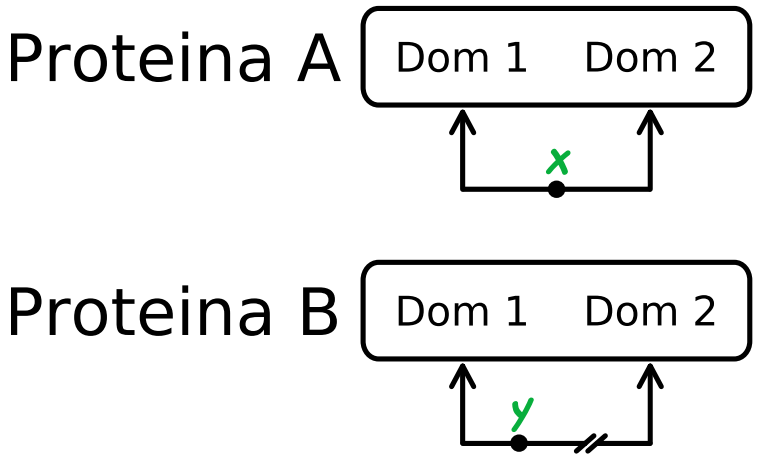


Figura 1.6: Legami intra ed inter-molecolari.

Con la doppia linea continua di Figura 1.2c viene indicato un generico legame covalente, per cui valgono le stesse considerazioni fatte per i legami non covalenti. La differenza risiede nel fatto che questi ultimi sono reversibili mentre i primi non lo sono. L'unico modo di spezzare un legame covalente è tramite una esplicita *divisione*[4], espressa tramite la linea a zig-zag di Figura 1.2f.

Gli ultimi semplici tipi di reazione sono la *conversione stechiometrica*[5] indicata da una freccia piena (Figura 1.2d), la *produzione*[6] di specie molecolari (Figura 1.2e), la *trascrizione*[7] (Figura 1.2i), la *degradazione*[8] (Figura 1.2g).

[4]In realtà la rottura di un legame covalente implica la rottura della molecola stessa. Per non creare confusione questo processo verrà chiamato *divisione* del legame covalente.

[5]La trasformazione uno a uno di una specie molecolare in un'altra.

[6]L'apparizione di prodotti senza perdita di reagenti

[7]La trascrizione è il processo mediante il quale le informazioni contenute nel DNA vengono trascritte enzimaticamente in una molecola complementare di RNA.

[8]ll processo per il quale una molecola viene distrutta.

1.2.2 Contingenze

Con il termine *contingenza* si indicano tutti quei modificatori di comportamento delle interazioni viste finora: stimolazione, inibizione, necessità e catalisi[9]. Esse si possono riferire sia alle interazioni sia ad altre contingenze, per questo motivo non puntano a dei nodi ma a degli archi.

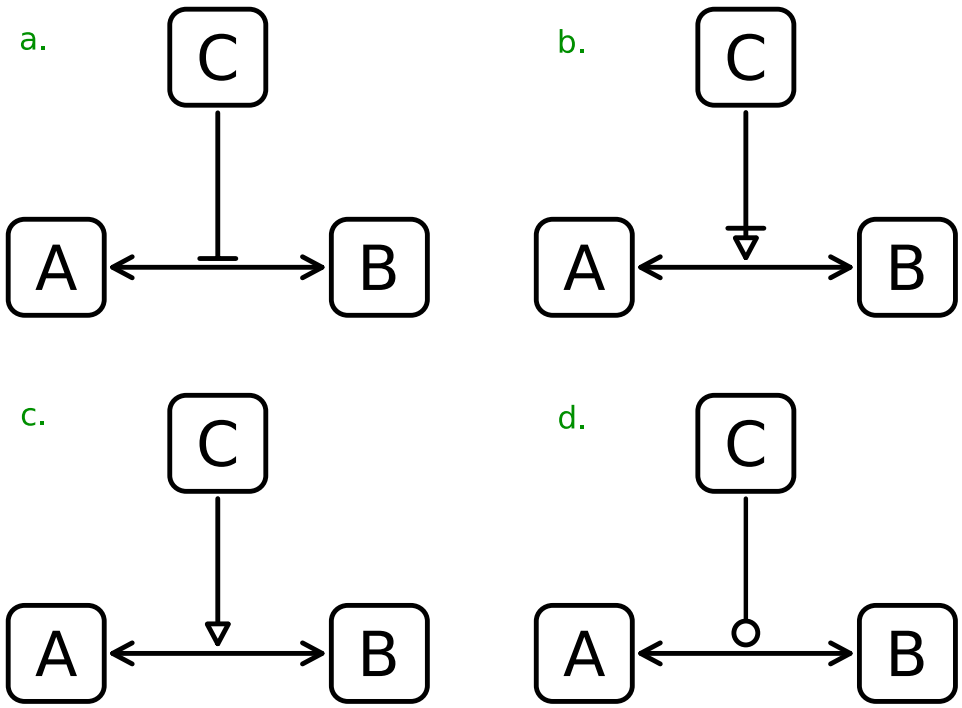


Figura 1.7: Simboli di contingenza: a. inibizione; b. necessità; c. stimolazione; d. catalisi.

In figura Figura 1.7 si possono vedere i simboli usati per le contingenze. Il primo è quello di *inibizione*, il cui effetto è quello di bloccare l'interazione a cui punta. Nella Figura 1.7a quindi il legame tra A e B è possibile se e solo se la molecola C non è presente. Il secondo simbolo (Figura 1.7b) è quello di *necessità* ed ha un significato speculare a quello di inibizione: il legame tra A e B è possibile solo in presenza della molecola C. L'effetto generato dal simbolo di *stimolazione* (Figura 1.7c) è quello di incoraggiare il legame tra A e B ed

[9]La catalisi è un fenomeno chimico attraverso il quale la velocità di una reazione chimica viene cambiata drasticamente per l'intervento di una sostanza, detta catalizzatore, che non viene consumata dal procedere della reazione stessa.

allo stesso tempo scoraggiarne la rottura. Il concetto di stimolazione va pensato in termini di velocità e probabilità associate ad ogni reazione: se stimolata una interazione ha una probabilità maggiore di avvenire. Si noti che questo ed il simbolo di produzione senza perdita di reagenti sono identici: quando la freccia vuota punta ad una specie molecolare è una produzione, quando punta ad una contingenza è una stimolazione. L'ultimo simbolo è quello di *catalisi* e incoraggia sia la creazione che la rottura del legame. L'interpretazione biologica della mappa di Figura 1.7d dice che C abbassa la barriera energetica dell'interazione tra A e B.

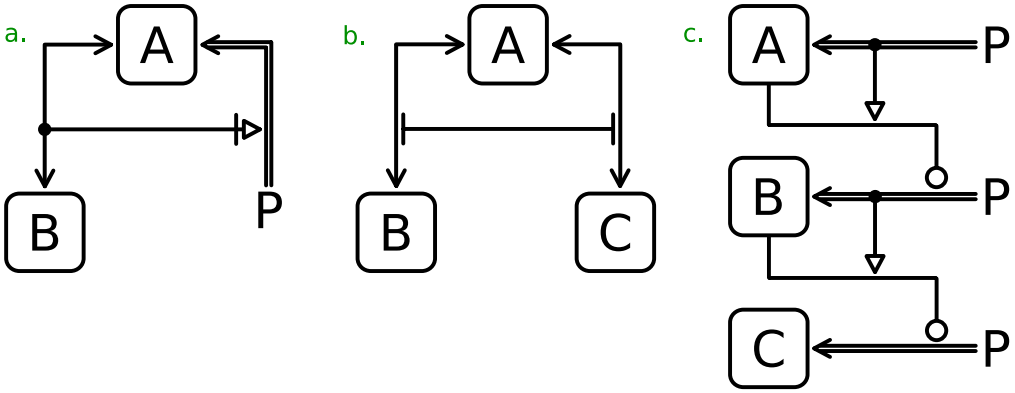


Figura 1.8: Esempi d'utilizzo dei simboli di contingenza: a. legami sequenziali; b. mutua inibizione; c. cascata di fosforilazione delle protein chinasi

Definizione 1.2.4 *Una contingenza è un modificatore del comportamento di una interazione da parte di una seconda interazione o di una specie molecolare.*

Tramite questi simboli è possibile modellare facilmente meccanismi anche molto complessi, in Figura 1.8 si possono vedere alcuni esempi.

La prima mappa (Figura 1.8a) esprime in un certo senso la sequenzialità di due eventi: affinchè possa avvenire la fosforilazione A deve legarsi con B, quindi deve prima avvenire la complessazione e poi la fosforilazione.

La seconda MIM (Figura 1.8b) indica una mutua inibizione che esprime il fatto che quei due legami non possono coesistere: appena una delle due molecole si lega ad A l'altra reazione viene inibita. Si noti che è stata utilizzata una abbreviaizone della notazione, collassando due inibizioni in un unico simbolo

ed eliminando i due nodi di partenza.

Nell'ultima (Figura 1.8c) viene rappresentata una cascata di fosforilazione delle protein chinasi: A e B sono protein chinasi attivate tramite fosforilazione. La molecola pA attiva pB la quale a sua volta attiva la fosforilazione di C.

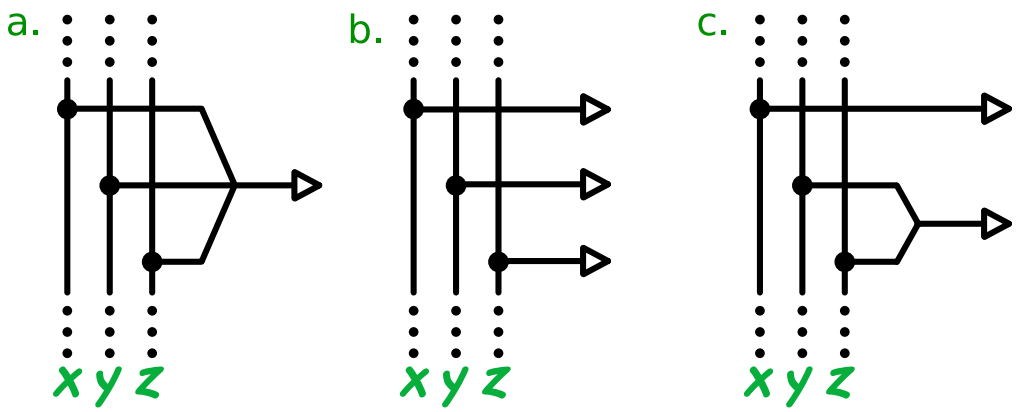


Figura 1.9: Contingenze multiple: a. x AND y AND z; b. x OR y OR z; c. x OR y AND z.

Le contingenze possono essere multiple. Il primo e più ovvio significato è quello per cui da un singolo nodo possono partirne diverse. Oltre ciò è possibile definire una combinazione di entità che attiva una contingenza. I tre frammenti di mappa di Figura 1.9 esprimono rispettivamente: x e y e z sono necessari per la stimolazione, x oppure y oppure z stimolano in maniera indipendente, x da sola stimola oppure y e z assieme stimolano.

1.3 Interpretazioni delle mappe

Uno degli aspetti che rende più interessanti le Mappe di Interazione Molecolare, sono le *interpretazioni* che è possibile dar loro. Nella prima interpretazione, chiamata *esplicita*, le interazioni coinvolgono esclusivamente gli elementi direttamente connessi ai suoi estremi. Al contrario, nell'interpretazione *combinatoria*, le interazioni rappresentano una specie di connessione funzionale che, se non diversamente specificato, non dipende dalle eventuali modificazioni o legami delle specie direttamente connesse.

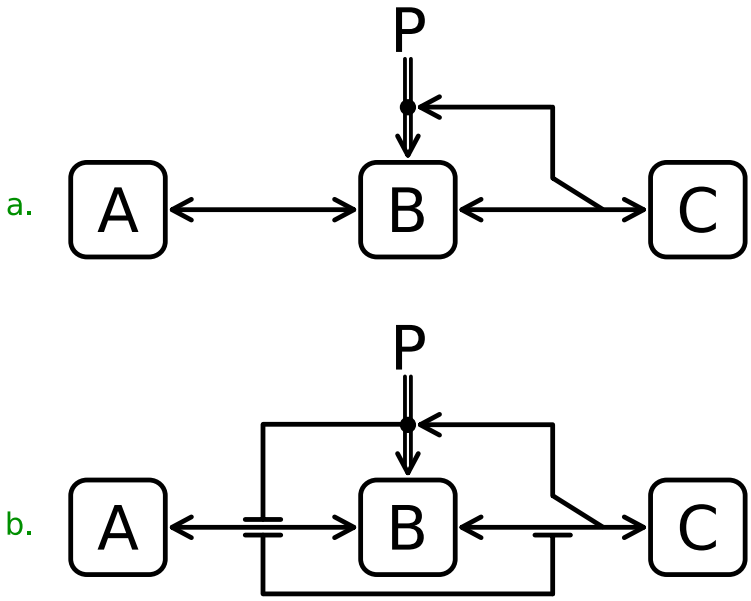


Figura 1.10: Esempio di come sia possibile limitare una MIM per far coincidere le due interpretazioni (si veda la Tabella 1.1).

Si coglie immediatamente la differenza guardando la mappa di Figura 1.10a, la quale contiene tre semplici specie molecolari elementari A, B e C, tre legami non covalenti ed una fosforilazione.

L'interpretazione *esplicita* obbliga le reazioni ad interagire con le sole specie direttamente connesse, quindi, come è riassunto in Tabella 1.1, i possibili complessi che si possono formare sono $A : B$, $B : C$, pB e $pB : C$.

Interpretando combinatorialmente invece, le reazioni non si curano delle modificazioni dei reagenti connessi permettendo ad esempio il legame tra A e B quando B è fosforilata, creando dunque $A : pB$. Questo complesso a sua volta si può legare a C dando origine a $A : pB : C$ e così via per tutte le possibili combinazioni.

Rielaborando le definizioni si può riassumere che nell'interpretazione esplicita può accadere solo ciò che è disegnato, mentre nella combinatoria può accadere tutto ciò che non è vietato.

Si capisce che abbracciare l'uno o l'altro punto di vista può originare significati molto diversi per la stessa mappa, come accade già per i pochi simboli in Figura 1.10a. Per fortuna è possibile, almeno in linea di principio, ricondurre una

mappa interpretata esplicitamente ad una, con stesso significato, interpretata combinatorialmente e viceversa. Il costo di queste operazioni è semplicemente l'aggiunta di simboli: una mappa esplicita avrà bisogno di un numero maggiore di simboli di reazione per poter esprimere tutti i composti ottenibili con l'altra interpretazione, viceversa, per limitare una mappa combinatoria è necessario introdurre simboli di contingenza per circoscrivere i comportamenti espressi a quelli desiderati.

La Figura 1.10b è un chiaro esempio di come sia possibile limitare una mappa in modo che i significati espressi dalle due interpretazioni coincidano, come si può vedere in Tabella 1.1.

Complessi	Figura 1.10a		Figura 1.10b	
	Espl.	Comb.	Espl.	Comb.
A:B	✓	✓	✓	✓
B:C	✓	✓	✓	✓
pB:C	✓	✓	✓	✓
A:pB		✓		
A:B:C		✓		
A:pB:C		✓		

Tabella 1.1: Differenti interpretazioni delle MIM di Figura 1.10

La terza ed ultima interpretazione è quella *euristica* che in un certo senso si colloca in mezzo tra le due appena discusse. Se nella esplicita non ci sono possibilità di interazioni al di fuori di quelle specificate e nell'altra di fatto ci sono tutte le combinazioni, quest'ultime nella euristica *potrebbero esistere*, nel senso che chi scrive la mappa decide a propria discrezione se ci sono o meno.

Essa è utilizzata principalmente dai biologi come metodo di organizzazione della conoscenza per annotare quanto si conosce, mettendo in evidenza quello che ancora rimane da scoprire. Non essendoci una definizione rigorosa al riguardo, risulta inutile ai fini di una simulazione automatica e per questo motivo non è interessante per gli scopi di questa tesi.

1.3.1 Combinatoria o esplicita?

Adottare l'interpretazione combinatoria durante la stesura di una MIM obbliga a fare particolare attenzione all'esplosione del numero dei possibili reagenti coinvolti in ogni reazione. Si capisce che quest'ultima è definita come una

connessione funzionale se si pensa ai reagenti come tutto ciò che può essere originato ai suoi due estremi. Ogni volta che si introduce una reazione infatti si sta aggiungendo in realtà una intera famiglia di reagenti: se si immettesse per esempio una nuova molecola D nella mappa di Figura 1.10a legata tramite una complessazione a C, si dovrebbero considerare tutti quei nuovi composti in cui C è legata anche alla molecola D, e precisamente: $B : C : D$, $pB : C : D$, $A : B : C : D$ ed $A : pB : C : D$.

Calcolare l'insieme delle combinazioni aggiunte introducendo una nuova reazione è possibile semplicemente guardando la mappa. Si potrebbe addirittura definire una sorta di proprietà transitiva dei costrutti grazie alla quale enumerare gli elementi di questo insieme. Alcuni costrutti come la complessazione o la fosforilazione verificherebbero questa proprietà, altri come la conversione o la degradazione no. Sebbene in linea di principio possibile, questo lavoro potrebbe risultare di ben poca utilità. Con il crescere della mappa infatti, ben presto il numero delle reazioni, proporzionale al numero di reagenti, diventa enorme.

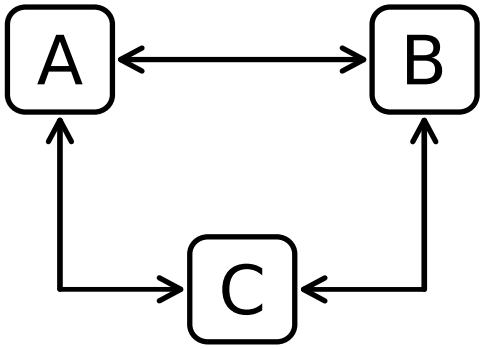


Figura 1.11: MIM con profonde differenze nelle due interpretazioni.

Questa loro crescita combinatoria permette di sintetizzare in pochi simboli un gran numero di possibili composti e reazioni, rappresentando sicuramente uno dei punti forti di questa interpretazione. Inoltre ci sono dei comportamenti biologici che non sono modellabili se non adottando questa interpretazione. Un esempio è dato in Figura 1.11: il significato esplicito è ovvio, quello combinatorio aggiunge la possibilità di creazione di lunghe catene del tipo $A : B : C : A : B : ...$ o perfino anelli, severamente proibite altrimenti.

D'altra parte, se ci si prefigge di simulare tramite il computer questa interpretazione, è necessario adottare delle tecniche in grado di fronteggiare una tale crescita esponenziale. Da questo punto di vista una mappa esplicita è sicuramente più semplice da gestire: come dimostrato in diverse occasioni ([19], [21]) è sufficiente pensare ad una o due reazioni chimiche per ogni simbolo di interazione (la complessazione per esempio nasconde sempre una decomplessazione, essendo il legame reversibile) e definire un sistema di equazioni differenziali da risolvere numericamente.

Come analizzato in precedenza, il prezzo da pagare consiste principalmente nel dover assumere le quantità delle molecole in gioco come continue oltre che dover sottostare ad un rigido determinismo del sistema, entrambi fattori che non trovano riscontro nelle entità biologiche che il metodo si prefigge di modellare, senza dimenticare il costo di stesura delle equazioni, direttamente proporzionale al numero di interazioni.

Questa tesi si prefigge l'obiettivo di sfruttare le potenzialità appena discusse dell'interpretazione combinatoria, senza dover pagare il prezzo della enumerazione esaustiva delle possibili reazioni. Come vedremo infatti, verrà fornita una codifica che a tutti gli effetti mantiene implicita tale lista, riuscendo comunque a tener conto ed eseguire ognuna di esse.

1.4 Ambiguità intrinseche della notazione

La notazione delle Mappe di Interazione Molecolare, per come è definita, presenta alcune ambiguità. Il lato positivo viene dalla possibilità di costruire delle mappe diverse aventi lo stesso significato, soprattutto adottando la notazione combinatoria.

Tornando alla Figura 1.10a si vede come il complesso $pB : C$ sia ottenibile sfruttando le due reazioni di complessazione, quella tra B e C e quella tra pB e C. Il primo caso sfrutta le proprietà combinatorie, il secondo vale anche nella interpretazione esplicita. Questo aspetto conferisce ulteriore flessibilità alla notazione, lasciando a chi la utilizza la libertà di scegliere la modalità di rappresentazione del fenomeno studiato a lui più congeniale.

Ci sono tuttavia dei casi in cui la la sintassi non definisce esplicitamente come si devono comportare i costrutti e quali dipendenze esistono tra loro, facendo emergere dei problemi.

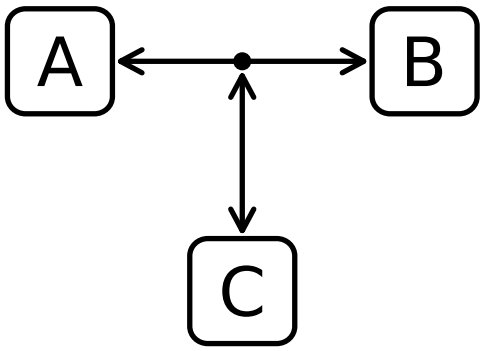


Figura 1.12: Questa semplicissima MIM nasconde una ambiguità.

A maggior ragione, se ci si prefigge l'obiettivo di automatizzare la simulazione dei comportamenti tenuti da una mappa, occorre assolutamente evitare casi di questo tipo, assicurandosi che essa si comporti in modo assolutamente deterministico.

Un semplice caso in cui questo non accade è illustrato in Figura 1.12, in cui la molecola A si lega al complesso $B : C$. Entrambe le reazioni sono reversibili, per cui si può presentare la situazione in cui all'interno del complesso $A : B : C$ si debba rompere il legame tra B e C, caso in cui non è chiaro che fine faccia A. La mappa infatti non specifica a quale delle due molecole si lega, quindi non si può sapere se essa rimarrebbe legata all'una o all'altra. Biologicamente questa mappa descrive il caso in cui per esempio il legame tra B e C modifica la forma o una delle due molecole stesse, in modo da attivare un sito sul quale si può legare A. Rimangono alcune domande: la rottura di $B : C$ comporta la scomparsa di questo sito? Oppure, è possibile che si rompa tale legame mentre la molecola A sta utilizzando il sito in questione?

Per questo motivo si parla di interpretazioni e non di semantiche, cercando di limitare le ambiguità avvicinandosi sempre più alla definizione di una vera e propria semantica formale.

Si noti inoltre che i casi che verranno presentati e risolti potrebbero non essere sufficienti ad assicurarsi la totale assenza di ambiguità. Il comune denominatore che le accomuna tuttavia è sempre lo stesso: ogni regola mira alla preservazione di un sito di interazione che sta venendo utilizzato e che potrebbe venir distrutto. Le regole presentate risolvono tutte le ambiguità che sono state incontrate nella stesura della presente tesi e possono fornire lo spunto e

la base per la soluzione di eventuali ambiguità che si presentino in mappe con una struttura particolare.

1.5 Regole di riscrittura delle mappe

La soluzione adottata per disambiguare le mappe è quella di assumere delle determinate ipotesi e via via aggiungere delle contingenze in modo da risolvere il problema. Questa aggiunta avviene per mezzo di regole di riscrittura che modificano le mappe nei punti dove sono presenti ambiguità, quindi utilizzando la notazione stessa. Sono state isolate due famiglie di regole, la prima focalizzata sulla soluzione delle ambiguità, la seconda per semplificare e rendere maggiormente trattabili le mappe.

Si noti che quest'ultimo gruppo di regole è più una comodità in fase di codifica che in fase di stesura delle mappe. Visto che il significato ultimo rimane inalterato, si può decidere di stendere le mappe usando l'intero insieme di costrutti e poi applicare una sorta di pre-processing automatico che le semplifichi.

Al momento queste regole devono venir applicate dal modellatore subito dopo aver disegnato la mappa. Tuttavia una delle evoluzioni che il lavoro della presente tesi potrebbe subire, consiste proprio nella loro applicazione automatica. Basandosi su una formalizzazione più completa della semantica delle mappe si potrebbero formalizzare anche queste regole in modo rigoroso, in modo da poter essere applicate come una specie di passo di *pre-processing*.

1.5.1 Risolvere le ambiguità

Tornando all'esempio di Figura 1.12 la domanda che ci si pone è se sia possibile o meno che il legame tra B e C si rompa mentre A è loro legata. Sembra biologicamente ragionevole assumere che questo comportamento non sia possibile e, per palesare questa tesi, si aggiungono alcune contingenze, nella fattispecie delle inibizioni, con lo scopo di impedire che il legame tra B e C si rompa se A è legata. La semplice regola è illustrata in Figura 1.13a.

Si presenta lo stesso problema in molteplici altri casi: la conversione stechiometrica di Figura 1.13b, che si assume non possa avvenire se il reagente è legato ad una terza molecola, per questo essa viene inibita dal legame di complessazione. Esattamente la stessa situazione si presenta in Figura 1.13c con una degradazione, ed allo stesso modo viene anch'essa inibita.

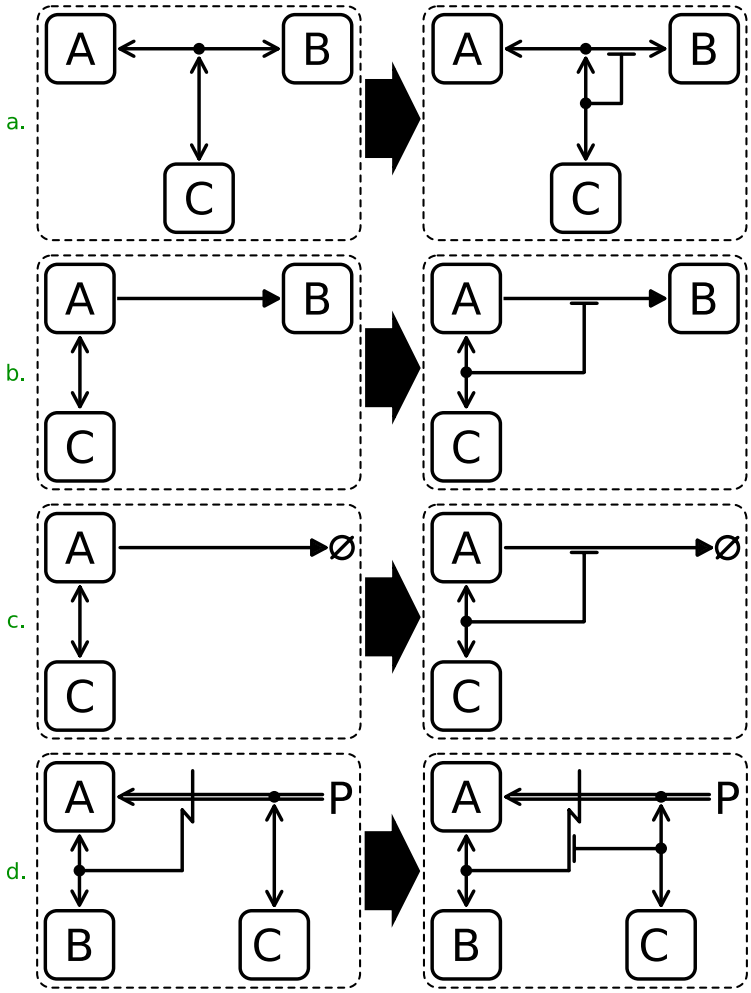


Figura 1.13: Regole di riscrittura delle MIM per risolvere le ambiguità.

Il caso illustrato in Figura 1.13d è leggermente più complicato: la molecola A può essere fosforilata o legarsi alla molecola B; quando fosforilata può legarsi anche a C. Il legame tra A e B determina la possibile divisione della particella di fosforilazione dalla molecola. Nel caso in cui pA sia legata con C, bisogna inibire la suddetta divisione, come mostrato dalla regola di riscrittura.

Il filo conduttore che lega i casi appena presentati consiste principalmente nella preservazione delle porte legate. Riguardando gli esempi infatti, ci si accorge che le reazioni analizzate possono rimuovere delle porte impegnate e di conseguenza compromettere il legame che queste ultime mantengono. La direzione verso la quale vanno le regole di riscrittura, è quella della preservazione

di tali legami, per evitare in seguito problemi di comportamenti inaspettati a livello di simulazione.

1.5.2 Regole di semplificazione delle mappe

Sfruttando la ridondanza della notazione è possibile ottenere gli stessi comportamenti da mappe diverse. Tale proprietà risulta molto utile quando si debbano codificare le mappe in un linguaggio di programmazione. Chiaramente meno costrutti si hanno da tradurre e più semplice sarà il codice. Inoltre a seconda del criterio scelto per la codifica, alcuni simboli potrebbero risultare di difficile traduzione, complicando magari tutto il sistema.

Tenendo ben presente questi scopi, si sono introdotte delle regole di semplificazione delle mappe grazie alle quali è possibile eliminare completamente alcuni simboli.

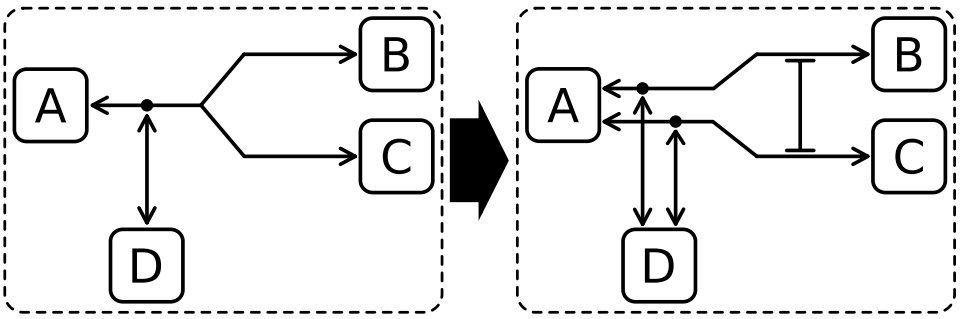


Figura 1.14: Rimozione del simbolo di competizione esclusiva per un sito di interazione.

Il primo caso è quello rappresentato dalla competizione esclusiva per una certa reazione, resa nelle mappe grazie all'unione in un angolo acuto di più simboli di reazione. Nell'esempio di Figura 1.14 si vede come B e C competano per legarsi ad A. Per eliminare la notazione ad angolo acuto è possibile sdoppiare il punto per cui competono le due molecole, ed inibire mutuamente i due legami. In questo caso l'effetto ottenuto è lo stesso: se è B a legarsi l'altra complessazione viene automaticamente inibita e viceversa. Bisogna sdoppiare anche il nodo $A : B/A : C$ in due nodi distinti, che andranno ad essere i punti di legame con la molecola D. Semplicemente sdoppiando il punto di contatto con la molecola per cui si compete, è possibile eliminare dalla mappa la notazione ad angolo acuto, ottenendo delle semplici complessazioni ed inibizioni.

Questa regola permette la formulazione di questa importante proprietà: ogni porta si lega esclusivamente ad una unica altra porta. Di conseguenza risulta possibile identificare univocamente ogni reazione semplicemente guardando la coppia di porte che le compete. Questo risulterà utile più avanti, quando si affronterà il problema di decidere se due descrizioni di complessi rappresentano in verità uno stesso complesso.

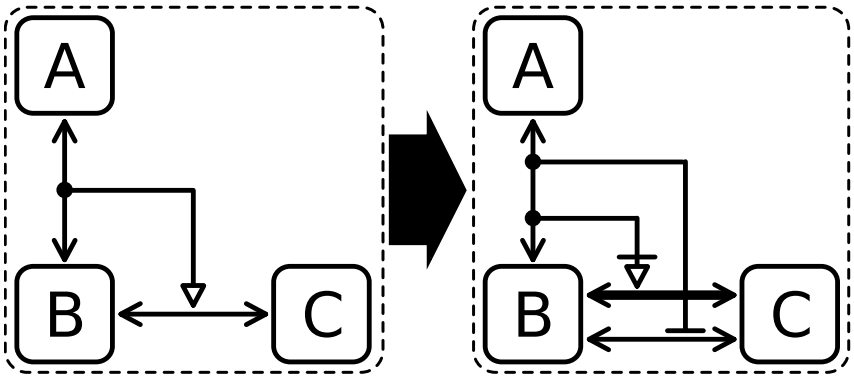


Figura 1.15: Rimozione del simbolo di stimolazione.

In maniera ancora più semplice è possibile eliminare il simbolo di stimolazione. Il suo effetto su una reazione è quella di aumentarne la velocità o la probabilità che essa accada. Anche qui basta sdoppiare la reazione stimolata, definendone una con un rate alto e l'altra con un rate basso e porre una necessità sulla prima ed una inibizione sulla seconda, come si vede in Figura 1.15. In questo modo in presenza della molecola A, la reazione veloce (quella in grassetto) viene attivata mentre l'altra inibita. Viceversa se A non è presente nel complesso, il legame veloce non può accadere mentre quello lento sì. Grazie a questa semplice riscrittura del grafo è possibile eliminare completamente il simbolo di stimolazione.

Un caso più complicato di doppia stimolazione si presenta quando si descrive in una mappa il legame cooperativo tra molecole, come accade in Figura 1.16. Quello che accade è che i due legami si stimolano a vicenda, appena uno dei due avviene l'altro è aiutato dal primo. La regola prevede di raddoppiare il legame stimolato e di far partire due simboli di inibizione e necessità dal legame stimolante. A prima vista la traduzione in figura potrebbe sembrare complicata, ma appena si guarda un po' meglio si vede che è stata semplicemente applicata questa regola: sono state sdoppiati i due legami stimolati e poste 2

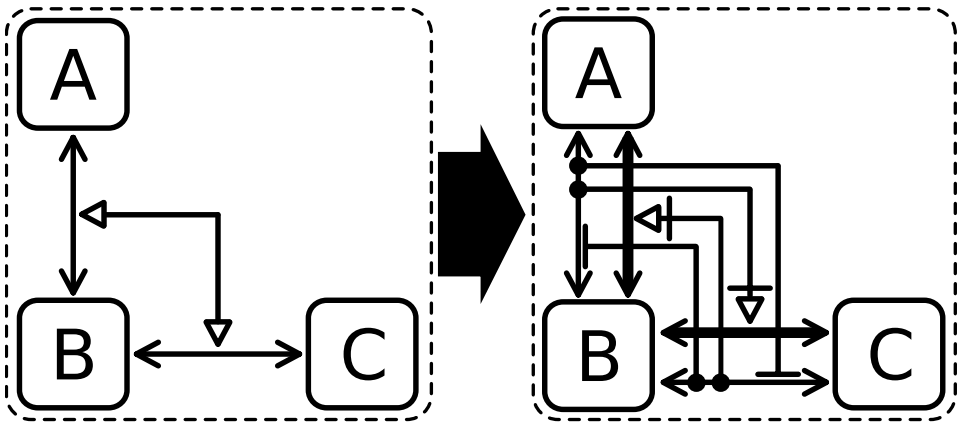


Figura 1.16: Rimozione del legame cooperativo tramite doppia stimolazione.

contingenze per ogni legame stimolante, in totale 4. La situazione di partenza è quella in cui sono possibili i soli legami lenti, i quali si inibiscono a vicenda e sono necessari affinchè possano avvenire quelli veloci. Dunque appena uno dei due avviene, automaticamente l'altra molecola verrà legata grazie alla complessazione veloce.

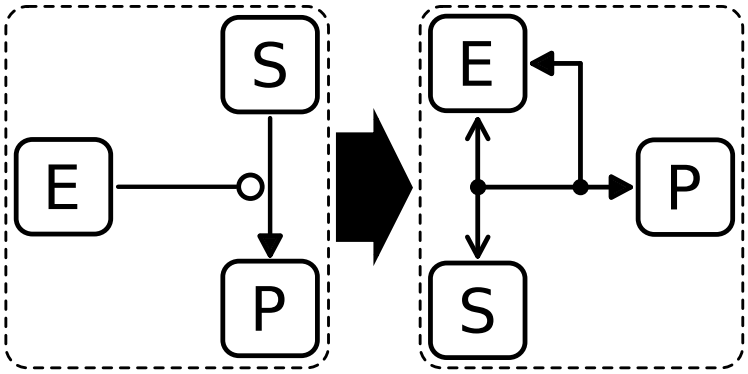


Figura 1.17: Rimozione del simbolo di catalisi.

Nell'esempio di Figura 1.17 è illustrata una semplificazione suggerita dagli autori stessi della notazione. Si tratta della stimolazione enzimatica di E nei confronti della conversione di S in P, e viene tradotta in una mappa che esplicita quello che accade: l'enzima E si lega al reagente S ed assieme si convertono nel prodotto P e di nuovo nell'enzima E. Molto simile è la regola di Figura 1.18 in cui l'enzima stimola la complessazione tra A e B. La riscrittura prevede che l'enzima si leghi ad una delle due molecole in modo da permettere

il legame tra di esse. Una volta avvenuto, questo legame stimola la rottura del legame tra enzima e complesso. Una piccola variazione di questa regola si ottiene se al posto di B ci si mette una particella di fosforilazione che si lega ad A, la riscrittura risulta identica.

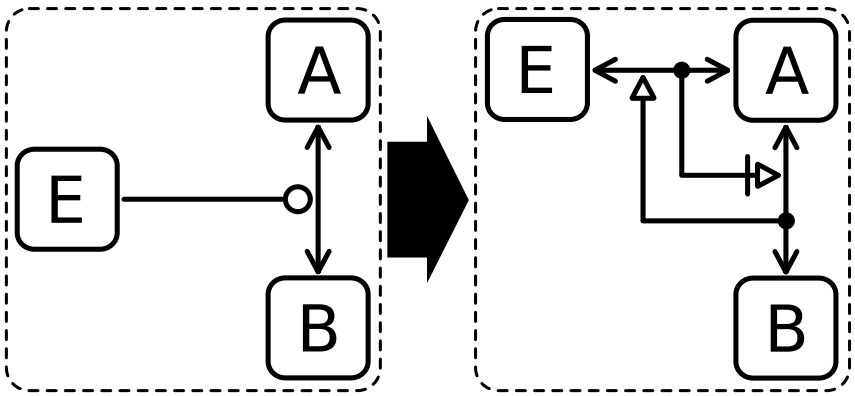


Figura 1.18: Altro esempio di rimozione del simbolo di catalisi.

Capitolo 2

Stochastic Concurrent Constraint Programming (sCCP)

sCCP ([3], [4]) è una estensione stocastica di CCP ([27]), il noto linguaggio di programmazione concorrente a vincoli. Tale estensione è ottenuta semplicemente considerando che ogni istruzione che interagisce con il constraint store (`ask` e `tell`) abbia una durata stocastica, calcolata tramite un valore casuale preso da una distribuzione esponenziale, il cui rate è funzione della configurazione corrente del constraint store, quindi dell'intero sistema.

In questo capitolo viene data una necessariamente breve descrizione di *CCP*, per poi introdurre altrettanto brevemente le *catene di Markov* alla base del modello semantico di riferimento dell'estensione stocastica di sCCP, verrà presentato il metainterprete utilizzato ai fini della simulazione in modo da permettere una più semplice comprensione di determinate scelte relative all'implementazione.

2.1 Concurrent Constraint Programming

Concurrent Constraint Programmin (CCP) è un linguaggio di programmazione per sistemi concorrenti che combina la sintassi di una algebra di processo con la potenza espressiva e computazionale dei vincoli logici [27]. Infatti CCP prevede che la memoria sia in realtà un contenitore di vincoli sulle variabili in gioco, chiamato *store*, i quali evolvono in maniera monotona: i vincoli possono venir aggiunti ma non rimossi. Per sfruttare questa particolarità di solito ad ogni variabile viene associato un intervallo di valori, che viene ristretto e

$Program ::= Decl.A$
$Decl ::= \epsilon \mid Decl.Decl \mid p(\overrightarrow{x}) : -A$
$G ::= \text{tell(c)} \mid \text{ask(c)}$
$M ::= G \leftarrow A \mid M + M$
$A ::= \mathbf{0} \mid A.A \mid M \mid A \parallel A \mid \exists_x A \mid p(\overrightarrow{x})$

Tabella 2.1: La sintassi di CCP.

raffinato tramite l'aggiunta di vincoli aggiuntivi durante la computazione.

In Tabella 2.1 è possibile vedere la sintassi di CCP. Si consigliano [27] e [28] a chi volesse conoscere i dettagli formali che definiscono il linguaggio: partendo da una relazione di *entailment* definita sui vincoli, passando per un reticolo algebrico completo per definire il *constraint system* usato poi come parametro per definirne la semantica operazionale.

Per gli scopi di questa tesi è sufficiente sapere che l'interazione con il constraint store è possibile solo per mezzo delle istruzioni `ask(c)`, la quale verifica se la configurazione attuale del constraint store verifica il vincolo c, e `tell(c)` che aggiunge un vincolo al constraint store. Si noti che queste istruzioni possono venir utilizzate al fine di sincronizzare processi: grazie all'adozione di una semantica bloccante per `ask`, viene infatti utilizzata come guardia per gli agenti che devono aspettare che un certo vincolo sia soddisfatto per venir eseguiti.

Il linguaggio inoltre offre la composizione sequenziale e parallela di agenti (operatore '.' ed operatore '||'), un meccanismo di scelta non deterministica (operatore '+') tra agenti dotati di guardia ed infine un operatore per la definizione di variabili locali (tramite $\exists_x$).

Il principale meccanismo utilizzato per memorizzare dati che cambiano nel tempo all'interno del constraint store sono le *stream variables*. Funziona in maniera piuttosto semplice: basti immaginare una semplice lista che all'inizio solo due elementi, il valore attuale da memorizzare ed una variabile non instanziata in coda. Ogni qual volta sia necessario aggiornarlo si aggiunge il nuovo valore rimpiazzando la variabile non instanziata con una lista di due elementi: il primo contiene il valore attuale, il secondo una variabile non instanziata.

Grazie a questo stratagemma l'ultimo elemento non nullo della lista contiene sempre il valore attuale della variabile.

2.2 Catene di Markov a tempo continuo

Una interessante famiglia di processi stocastici discreti sono i processi *Markoviani*. Essi hanno la particolarità di essere privi di memoria, cioè il loro comportamento è definito unicamente in termini dello stato attuale e non degli stati passati. Nel caso lo spazio degli stati sia finito o comunque abbia un numero finito di elementi, il processo prende il nome di *catena di Markov*.

Definizione 2.2.1 *Una* catena di Markov a tempo continuo *(CTMC) è un particolare processo casuale a tempo continuo, continuo a destra, per cui vale la proprietà di assenza di memoria. Cioè per ogni n, t_i, s_i vale:*

$$\mathbb{P}(X_{t_n} = s_n \mid X_{t_0} = s_0, \ldots, X_{t_{n-1}} = s_{n-1}) = \mathbb{P}(X_{t_n} = s_n \mid X_{t_{n-1}} = s_{n-1}).$$

È possibile descrivere una CTMC usando la sua *matrice generatrice* Q, avente una riga ed una colonna per ogni elemento dell'insieme degli stati S. Gli elementi rispettano la proprietà

$$q_{ij} \geq 0, i \neq j$$

$$q_{ii} = -\sum_{j \neq i} q_{ij}$$

in modo che ogni riga sommi a 0.
Un altro modo intuitivo per pensare alle catene di Markov è visualizzarle tramite un grafo completo con un nodo per ogni elemento di S e con gli archi etichettati dal corrispondente elemento di Q.

La proprietà di Markov relativa all'assenza di memoria può essere equivalentemente descritta tramite una competizione stocastica. Si associa una variabile casuale esponenzialmente distribuita $T_{ij} \sim Exp(q_{ij})$[1] ad ogni elemento $q_{ij} \geq 0$ di Q. Quando la catena è nello stato s_i, si inizia una competizione stocastica

[1]Una variabile casuale $T : \Omega \to [0, \infty]$ ha una *distribuzione esponenziale* di parametro (o rate) λ (indicato con $T \sim Exp(\lambda)$), con $\lambda \in [0, \infty)$, se la sua funzione di probabilità è

$$\mathbb{P}(T > t) = e^{-\lambda t} \text{ per ogni } t \geq 0$$

Si rimanda a [4] per maggiori dettagli a riguardo.

tra tutte le variabili casuali T_{ij} per cui $s_j \in S, q_{ij} \geq 0$. La competizione è vinta dalla variabili più veloce, cioè quella che realizza $T_i = \inf_{s_j \in S}\{T_{ij}\}$. Se la vincitrice è T_{ik} allora il sistema si posta in s_k in un tempo $T = T_{ik}$, stato nel quale partirà la prossima competizione tra le variabili casuali di questo stato.

È possibile descrivere una catena di Markov a tempo continuo in termini di *jump chain* e *holding times*. Intuitivamente i *jump times* della catena sono definiti come gli istanti in cui essa cambia stato. Grazie a loro è possibile ricavare, per differenza, gli holding times, cioè i tempi di permanenza della catena in ogni stato. Eliminando l'informazione temporale dalla catena di Markov si definisce la jump chain: la sequenza di stati che la catena attraverserà.

È possibile dimostrare che una jump chain $(Y_n)_{n\geq 0}$ è una catena di Markov a tempo discreto con matrice stocastica Π definita in termini di Q (la matrice generatrice della catena di partenza), avente elementi $\pi_{ij} = \frac{q_{ij}}{q_i}$.

Le caratterizzazioni sono equivalenti. Quest'ultima ci permette di vedere la scelta del prossimo stato attraverso una distribuzione di probabilità che normalizza i rate sugli archi uscenti $\frac{q_{ij}}{q_i}$, quindi collegando la competizione stocastica al grafo etichettato.

2.3 Sintassi di sCCP

Le istruzioni di sCCP sono ottenute partendo da quelle di CCP, aggiungendo dei *rate* λ alle istruzioni di ask e tell, quelle che interagiscono con lo store. Questi rate permettono una doppia interpretazione. La prima li tratta come fossero degli indici di priorità, la seconda come se esprimessero il tempo T richiesto per terminare l'interazione con lo store. Questa variabile T è continua e casuale, ottenuta tramite una distribuzione esponenziale con funzione di densità:

$$f(\tau) = \lambda e^{-\lambda\tau}$$

in cui λ è chiamato il rate della variabile esponenziale casuale, positivo e reale, e può essere intuitivamente visto come una sorta di frequenza attesa dell'istruzione associata.

Le funzioni associate alle istruzioni ask e tell sono dunque $\lambda : \mathbb{C} \to \mathbb{R}^+$, definite correttamente in quanto vengono calcolate a partire dallo stato del sistema (tramite lo store) ed assumono un valore reale e positivo.

$$Program = D.A$$

$$D = \varepsilon \mid D.D \mid p(\overrightarrow{x}) : -A$$

$$A = \mathbf{0} \mid \text{tell}_\infty(c).A \mid M \mid \exists_x A \mid A \parallel A$$
$$M = \pi.G \mid M + M$$
$$\pi = \text{tell}_\lambda(c) \mid \text{ask}_\lambda(c)$$
$$G = \mathbf{0} \mid \text{tell}_\infty(c).G \mid p(\overrightarrow{y}) \mid M \mid \exists_x G \mid G \parallel G$$

Tabella 2.2: Sintassi di sCCP.

La sintassi di sCCP è mostrata in Tabella 2.2. Un programma consiste in una lista di procedure ed una configurazione iniziale. Le procedure sono dichiarate specificando il loro nome ed i loro parametri formali, con una dichiarazione tipo $p(\overrightarrow{x}) : -A$, in cui si richiede che le variabili libere di A siano sottoinsieme delle variabili $\overrightarrow{x}$. Le ultime quattro righe della tabella illustrano come è possibile definire gli agenti. Con π vengono identificati ask e tell, che possono venire utilizzati come guardia per le istruzioni non stocastiche. In questo modo si evita il caso di una ricorsione che utilizza sole istruzioni immediate, quindi non di durata stocastica. Un agente A dunque può effettuare una scelta stocastica tra diverse azioni (M), effettuare un tell instantaneo di un certo vincolo ($tell_\infty(c)$), dichiarare variabili locali ($\exists_x A$) oltre ad essere combinato con altri agenti sequenzialmente (con .) o in parallelo (con $\parallel$). Una trattazione completa di sCCP, completa di una semantica operazionale può essere trovata in [4].

2.4 La semantica di sCCP

Si fornisce ora l'intuizione che sta alla base della semantica di sCCP. Per una trattazione completa e rigorosa si rimanda a [3].

La prima cosa da definire è lo stato in cui si trova il sistema. Esso dipende dallo *spazio dei processi* $\mathcal{P}$ che contiene tutti i possibili processi che possono apparire durante l'esecuzione di un programma. Inoltre dipende da $\mathcal{C}$, il *constraint system*[2]. utilizzato, in pratica lo spazio di tutti i possibili constraint

[2]i dettagli del reticolo completo che descrive il constraint system sono consultabili in [3]

store.

Dunque lo stato in cui è il sistema in ogni istante è un punto nello spazio $\mathbf{C} : \mathcal{P} \times \mathcal{C}$, indicato dalla coppia $\langle A, d \rangle$.

A questo punto è possibile definire una relazione di transizione istantanea $\rightarrow \subseteq \mathbf{C} \times \mathbf{C}$ grazie alla quale costruire la semantica operazionale dei costrutti che non hanno ingrediente stocastico. Per esempio il caso del tell istantaneo è definita dalla regola:

$$\langle tell_{\infty}(c), d \rangle \ \rightarrow \ \langle \mathbf{0}, d \bigsqcup c \rangle$$

in cui $\bigsqcup$ è l'operazione di least upper bound del constraint system $\mathcal{C}$. In modo simile viene definita la semantica della chiamata a procedura e degli operatori di composizione sequenziale e parallela.

Si procede quindi con la definizione della semantica dei costrutti stocastici, definendo la relazione $\Longrightarrow \subseteq \mathbf{C} \times [0,1] \times \mathbb{R}^{+} \times \mathbf{C}$. Le due etichette associate ad ogni transizione stocastica sono la probabilità associata a questa transizione ed il suo rate globale.

Intuitivamente questa relazione descrive una CTMC in termini di jump chain ed holding times.

Per esempio la semantica dell'istruzione $ask_{\lambda}(c)$ è definita dalla regola:

$$\langle \mathrm{ask}_{\lambda}(c), d \rangle \ \Longrightarrow_{(1,\lambda(d))} \ \langle \mathbf{0}, d \rangle \quad se \ d \vdash c$$

dove $\lambda(d)$ è il rate dell'istruzione calcolato sullo stato attuale del constraint store. L'agente avanza con probabilità pari a 1 se e solo se $d \vdash c$, cioè se il constraint store verifica il vincolo c tramite la relazione di entailment $\vdash$.

Grazie a questa regola è possibile ricorsivamente definire la semantica degli agenti con guardia, π di Tabella 2.2:

$$\frac{\langle \pi, d \rangle \ \Longrightarrow_{(p,\lambda)} \ \langle \mathbf{0}, d' \rangle}{\langle \pi.A, d \rangle \ \Longrightarrow_{(p,\lambda)} \ \overrightarrow{\langle A, d' \rangle}}$$

dove appunto $\pi = \mathrm{ask}$ oppure $\pi = \mathrm{tell}$ e $\overrightarrow{\langle A, d' \rangle}$ è la chiusura transitiva dello stato $\langle A, d' \rangle$ rispetto alla relazione di transizione instantanea $\rightarrow$. In pratica la regola dice che se la guardia è verificata allora si applica finchè possibile $\rightarrow$ e si va nello stato risultante. Questo inoltre permette di affermare che

la prima istruzione dell'agente A sarà stocastica, avendo già consumato tutte quelle istantanee. Per i dettagli di questa e delle altre regole si rimanda a [3].

Una volta definite le regole della semantica operazionale è possibile definire un *labeled transition system*, un grafo etichettato che descrive il programma. Esso ha un nodo per ogni configurazione del sistema $\mathbf{C}$ e si aggiunge un arco tra due nodi ogni qual volta è possibile derivare una transizione tra le due configurazioni. L'etichetta connessa agli archi riporta le informazioni relative a probabilità e rate.

Infine, avendo a disposizione un grafo è finalmente possibile definire una catena di Markov continua per la quale il tempo di permanenza in uno stato, holding time, è dato da una distribuzione esponenziale il cui rate è la somma dei rate delle transizioni uscenti. Per questa catena si genera una possibile traccia (in pratica la si simula) tramite l'algoritmo di Gillespie, come si vedrà nei prossimi paragrafi.

2.5 Il metainterprete

Nei precedenti paragrafi sono state date per valide parecchie assunzioni sul linguaggio, come per esempio le funzionalità del constraint store. È necessario specificare come esse vengono implementate al fine di capire meglio la codifica data per le MIM.

La maggior parte del lavoro in tal senso è delegato ai meccanismi già sviluppati nell'ambito di CCP, quindi primariamente la programmazione a vincoli logici (CLP). Un modo efficace per definire i vincoli è definirli per mezzo del motore inferenziale di Prolog, il quale grazie a tecniche di unificazione, backtracking, soluzione, ecc. generano con il minimo sforzo ed in modo implicito i vincoli necessari. Per i dettagli su tali meccanismi si rimanda a [14] e [15].

Grazie alla scelta di Prolog è possibile definire il metainterprete per sCCP in modo semplice, sfruttando la potenza e semplicità di gestione dell'agoritmo di unificazione. Inoltre offre notevoli vantaggi anche in termini di librerie disponibili, in grado di gestire automaticamente operazioni complicate come l'*entailment* di un vincolo. Per l'implementazione del metainterprete sCCP è stato scelto SICSTus Prolog [29] per le numerose librerie offerte e per la sua efficienza.

Come detto il principale meccanismo utilizzato per memorizzare dati che varia-

no nel tempo all'interno del constraint store sono le *stream variables*. Tuttavia in pratica l'implementazione è resa con un `retract` del vecchio vincolo ed un `assert` del nuovo valore.

Un aspetto del metainterprete di cui vale la pena discutere è di certo la gestione dei rate associati agli agenti. Come detto sCCP offre la possibilità di formulare tali rate come funzione del constraint store. Questo è ottenuto grazie ad un meccanismo per il quale si definisce un predicato che li calcola (nelle prossime sezioni si vedrà in che modo), si passa il nome di tale predicato al metainterprete assieme ai parametri con il quale calcolarlo ed il motore di simulazione si occupa del resto. Grazie a ciò è possibile definire il calcolo dei rate in modo parametrico e tutto sommato semplice: la potenza espressiva a disposizione per il suo calcolo è quella di Prolog stesso.

L'ultima nota riguarda la sintassi utilizzata per definire i programmi sCCP veri e propri. Per ragioni implementative essa è leggermente diversa da quella presentata nella Tabella 2.2. Comunque queste differenze sono tutt'altro che sostanziali, come si vedrà nei prossimi capitoli nei quali verranno presentati esempi di agenti sCCP.

2.5.1 Simulazione stocastica

Il motore di simulazione del metainterprete adotta l'algoritmo di Gillespie ([12], [13]) per generare una possibile traccia di una catena di Markov generata a partire dal programma sCCP. Essenzialmente vengono generate delle tracce di una CTMC, sfruttando la sua caratterizzazione in termini di Jump Chain e Holding Times.

Una tipica configurazione del metainterprete prevede di avere degli agenti in parallelo, pronti ad essere eseguiti. Per come è strutturato il linguaggio, vengono prima eseguite tutte le istruzioni istantanee per creare successivamente una somma stocastica di tutti gli altri agenti [4]. A questo punto si testano le guardie degli agenti per eliminare quelle inattive. Si sommano dunque i rate λ_i associati ad ognuno degli agenti rimasti per calcolare λ, il rate di uscita dallo stato corrente. Ora è possibile decretare il vincitore della competizione con probabilità $\frac{\lambda_i}{\lambda}$. Il tempo trascorso è dunque generato da una distribuzione esponenziale con rate λ.

Per i dettagli di questa procedura si rimanda a [4].

2.6 Un primo esempio di utilizzo di sCCP: le reazioni biochimiche

Con *reazioni biochimiche* si descrivono le usuali reazioni chimiche, i cui reagenti sono principalmente proteine. Esse si esprimono di solito nella forma:

$$R_1 + \ldots + R_n \rightarrow_k P_1 + \ldots + P_m$$

dove gli R_i sono i *reagenti* mentre P_i i *prodotti*. Questa relazione ci dice che combinando il giusto numero di reagenti è possibile ottenere quei prodotti. La reazione inoltre ha un rate associato k, che esprime fondamentalmente la sua velocità di base. Per ottenere la vera velocità di reazione, secondo il principio di *mass-action*[3], bisogna prendere in considerazione le quantità dei vari reagenti, se N_i è il numero di molecole R_i allora il rate reale è $k \cdot \prod_i N_i$.

$\text{reaction}(k, [R_1, \ldots, R_n],\ [P_1, \ldots, P_m]) :-$
$\quad \text{ask}_{r_{MA}(k,R_1,\ldots,R_n)} \left(\bigwedge_{i=1}^{n} (R_i > 0)\right).$
$\quad \big(\|_{i=i}^{n} \text{tell}_\infty(R_i = R_i - 1) \,\|\|_{j=1}^{m} \text{tell}_\infty(P_j = P_j + 1)\big).$
$\quad \text{reaction}(k, [R_1, \ldots, R_n],\ [P_1, \ldots, P_m])$
con: $\quad r_{MA}(k, X_1, \ldots, X_n) = k \cdot X_1 \cdots X_n$

Tabella 2.3: Possibile codifica di una reazione chimica in pseudo-codice sCCP.

In Tabella 2.3 si vede una possibile codifica in sCCP delle reazioni chimiche. L'agente per prima cosa si assicura che siano presenti nel sistema tutti i reagenti necessari (le quantità R_i siano maggiori di zero), successivamente modifica le quantità di reagenti (sottraendo uno) e prodotti (aggiungendo uno) per poi richiamare se stesso ricorsivamente.

Ogni qual volta si abbia bisogno specificare una diversa reazione basta istanziare in modo diverso l'agente presentato, modificando semplicemente i reagenti ed i prodotti.

Accoppiandone due in parallelo è possibile inoltre descrivere una reazione bidirezionale. Le due reazioni diretta e inversa

$$R_1 + \ldots + R_n \rightarrow_{k_1} P_1 + \ldots + P_m$$
$$P_1 + \ldots + P_m \rightarrow_{k_2} R_1 + \ldots + R_n$$

[3]esistono dinamiche diverse dalla mass-action, come per esempio quella di Michaelis-Menten o di Hill, in cui i rate vengono calcolati in modo diverso.

possono venir abbreviate in

$$R_1 + \ldots + R_n \rightleftharpoons_{k_1}^{k_2} P_1 + \ldots + P_m$$

e descritte in sCCP sfruttando due copie in parallelo dell'agente `reaction` di Tabella 2.3:

$$\text{reaction}(k_1, [R_1, \ldots, R_n],\ [P_1, \ldots, P_m]) \parallel$$
$$\text{reaction}(k_2, [P_1, \ldots, P_m],\ [R_1, \ldots, R_n])$$

in cui due agenti competono stocasticamente secondo i propri rate k_1 e k_2 per le due reazioni.

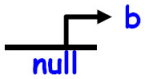
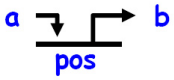
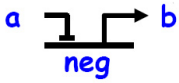
Questo stesso approccio verrà seguito per la codifica delle Mappe di Interazione Molecolare. Come si vedrà verrà creato un parallelo di agenti in competizione, ognuno associato ad una reazione regolata da una dinamica mass-action, quindi con rate proporzionale alle quantità di reagenti.

2.7 Un secondo esempio di utilizzo di sCCP: i gene gates

Grazie all'estensione stocastica di cui è equipaggiato sCCP è possibile modellare con successo diversi formalismi utilizzati nella biologia sistemica. Tra i meno conosciuti ma più innovativi ci sono i *gene gates* [2], utilizzati per descrivere appunto i geni ed i loro comportamenti.

All'interno di ogni cellula solo alcuni geni sono prodotti in un dato momento, per questo motivo una delle sue funzioni più importanti risulta quella della regolazione dell'espressione genica. Esistono delle proteine chiamate *fattori di trascrizione*, che si legano alla regione promotrice dei geni, la porzione di DNA immediatamente prima della regione codificatrice, con il compito di sopprimere o stimolare l'attività di trascrizione. Questi fattori di trascrizione sono a loro volta prodotti da geni, innescando un sistema in cui geni producono proteine che regolano altri geni, grazie a diverse reti di feedback positive e negative.

Il formalismo prevede tre tipi di gene gates: nullary, positive e negative gates che rispettivamente rappresentano geni con attività di trascrizione ma senza regolazione, geni le cui trascrizioni possono venir stimolate ed infine quelli le cui attività di trascrizione possono venir inibite. Il formalismo si pone ad un livello di astrazione per cui ogni gene produce direttamente la proteina invece che la molecola di mRNA. Queste proteine possono intervenire nella regolazione di altri geni o venir degradate, quindi scomparire.

	$\text{null\_gate}(k_p, X) :-$ $\text{tell}_{k_p}(X = X + 1).\text{null\_gate}(k_p, X)$
	$\text{pos\_gate}(k_p, k_e, k_f, X, Y) :-$ $\text{tell}_{k_p}(X = X + 1).\text{pos\_gate}(k_p, k_e, k_f, X, Y)$ $+\text{ask}_{r(k_e,Y)}(true).\text{tell}_{k_f}(X = X + 1).$ $\text{pos\_gate}(k_p, k_e, k_f, X, Y)$
	$\text{neg\_gate}(k_p, k_i, k_d, X, Y) :-$ $\text{tell}_{k_p}(X = X + 1).\text{neg\_gate}(k_p, k_i, k_d, X, Y)$ $+\text{ask}_{r(k_i,Y)}(true).\text{ask}_{k_d}(true).$ $\text{neg\_gate}(k_p, k_i, k_d, X, Y)$

con $r(k, Y) = k \cdot Y$.

Tabella 2.4: Possibile codifica dei Gene Gates in sCCP.

Una possibile codifica in sCCP di un formalismo di questo genere è molto semplice: si tiene traccia del numero e tipo di proteine presenti nel sistema all'interno del constraint store mentre i gene gates vengono modellati con gli agenti di Tabella 2.4. Il nullary gate è il più semplice: aumenta semplicemente il numero delle proteine che gli competono ad una certa velocità. Le positive gates possono produrre la propria proteina alla velocità base oppure entrare in uno stato eccitato che determina un aumento della velocità di produzione. Le negative gates sono del tutto simili: lavorano alla velocità normale oppure entrano in uno stato inibito che diminuisce la produzione. Questi gates entrano nello stato eccitato o inibito con probabilità proporzionale al numero di fattori di trascrizione presenti nel sistema.

Per esempio nell'agente pos_gate, c'è una competizione tra i due rate k_p (stato normale, rate fisso) e $r(k_e, Y) = k_e \cdot Y$ che porta allo stato eccitato, in cui k_e è appunto il rate nello stato eccitato ed Y è il numero di fattori di trascrizione che regolano questo gate. In entrambi i casi l'agente rilancia se stesso prima di terminare.

2.7.1 Circuito bistabile tramite gene gates

Un semplice circuito bistabile può essere rappresentato tramite due negative gates che si inibiscono a vicenda, come è visualizzato in Figura 2.1. Il primo produce la proteina A e viene represso dalla B, il secondo viceversa produce la B e viene regolato da A. Inoltre ci sono due reazioni di degradazione che eliminano A e B ad una velocità costante. La rete è bistabile nel senso che viene prodotta una sola proteina alla volta: se le concentrazioni iniziali sono entrambe zero il sistema parte con una fluttuazione stocastica finchè viene scelta la proteina su cui si stabilizzerà il sistema.

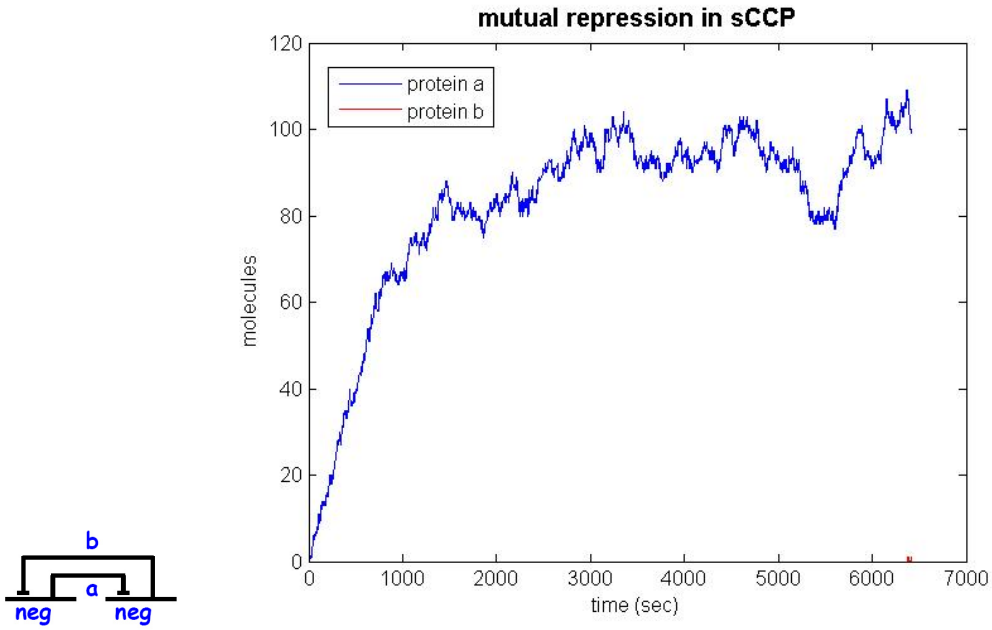


Figura 2.1: Circuito bistabile descritto in termini di Gene Gates e simulato tramite sCCP.

In Figura 2.1 si può vedere l'andamento della simulazione ottenuta grazie alla stesura delle poche righe di sCCP presenti in Tabella 2.4.

Capitolo 3

Codifica delle MIM in sCCP

Il linguaggio utilizzato per la codifica è Sictus Prolog [29], essendo l'unico per cui è stato scritto un interprete di sCCP. Essendo un linguaggio logico, bisogna necessariamente pensare in termini di vincoli e predicati. Questo risulta molto comodo in alcune fasi, in cui con il minimo sforzo si delegano compiti, anche complicati, al sistema di propagazione di vincoli che il linguaggio stesso offre.

Dunque scrivendo gli agenti sCCP è necessario tenere a mente che le istruzioni che essi eseguiranno saranno vincoli logici, i quali interagiranno con il constraint store, in stile Prolog. La codifica che viene presentata segue proprio questi principi, cercando di sfruttare al meglio le entità a se stanti come gli agenti e la potenza espressiva di Prolog per la definizione dei vincoli logici.

3.1 L'approccio

Come già argomentato, il problema principale quando si vuole simulare una mappa interpretata combinatorialmente è gestire i complessi. Elaborare una codifica in sCCP richiede pensare in termini di agenti, vincoli logici e predicati: viene naturale ipotizzare che ogni differente reazione possa venir tradotta in un tipo differente di agente ed instanziata secondo le necessità.

Questi agenti di reazione hanno bisogno di una descrizione dei complessi su cui poter lavorare: in ogni momento c'è bisogno di sapere se la reazione è possibile (se esistono tutti i reagenti e tutte le contingenze sono soddisfatte) e, se è il caso, eseguirla. Come si vedrà in seguito, il fulcro di questo approccio di codifica risiede proprio nella descrizione dei complessi: un grafo di nodi ed archi con un insieme finito di *porte*. Grazie alla sua flessibilità, il contrainst store memorizzerà e permetterà di gestire strutture dati non semplici quali i

grafi associati ai complessi. Tali strutture verranno modificate dagli agenti per mezzi di vincoli logici, quindi sfruttando le potenzialità offerte da sCCP, senza modificare i significati associati ai simboli delle Mappe di Interazione Molecolare.

Inoltre il potere espressivo del constraint store verrà naturalmente utilizzato per la codifica delle contingenze, associando loro dei vincoli logici che modificano la rappresentazione dei complessi, le strutture ad essi associate, secondo la semantica data dalle MIM.

3.2 Uso delle porte

Ogni arco di reazione delle MIM può, in linea di principio, rappresentare una intera famiglia di reazioni che coinvolgono un nutrito insieme di complessi diversi. Tuttavia la mappa è chiara a proposito dei reagenti: la reazione di complessazione, per esempio, utilizza esattamente due reagenti, quelli ai capi della freccia uncinata.

Questa freccia è collegata ai reagenti in due punti di interazione che chiamiamo *porte*. Esse descrivono i possibili modi tramite i quali le molecole possono legarsi, venir fosforilate o convertite oppure, più in generale, interagire. Ogni arco in una mappa parte e termina in una *porta* differente.

Nella mappa di Figura 3.1 si possono vedere tre legami non covalenti ed uno covalente che coinvolgono in tutto quattro molecole. Nella figura vengono assegnati esplicitamente dei nomi alle porte, P_1, P_2, ecc. Si può affermare che la specie molecolare A ha tre porte: P_1, P_2 e P_3; B ne ha due: P_4 e P_5; C e D ne hanno una sola, rispettivamente P_6 e P_7. Per chiarezza anche i tre simboli di complessazione hanno dei nomi: C_1 (tra A e B), C_2 (tra B e C) ed infine C_3 (tra A e D).

Ad ogni porta viene assegnato un nome differente, chiamato `port_id`, che la identifica inequivocabilmente all'interno di una MIM.

Diciamo che una porta è *libera* quando la molecola a cui appartiene è disponibile per la reazione a cui fa capo. Se la reazione corrispondente crea dei legami, a reazione avvenuta diciamo che la porta è *legata*: è occupata a mantenere il legame. Le porte libere possono essere *inibite*, cioè non sono impegnate in reazioni ma non sono comunque disponibili a farle avvenire, mo-

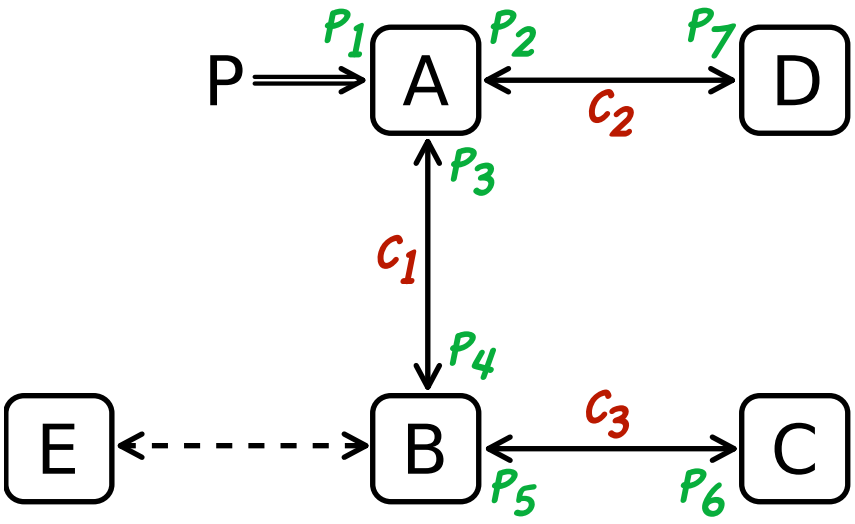


Figura 3.1: Una mappa nella quale sono stati assegnati dei nomi alle porte ($P_1, P_2, ...$) ed alle reazioni (C_1, C_2 e C_3).

dellando, per esempio, il caso in cui un certo legame inibisce un particolare sito di interazione.

$$\begin{array}{rcrcl} A & + & B & \rightarrow & A:B \\ A & + & B:C & \rightarrow & A:B:C \\ pA & + & B & \rightarrow & pA:B \\ pA & + & B:C & \rightarrow & pA:B:C \\ A:D & + & B & \rightarrow & A:D:B \\ A:D & + & B:C & \rightarrow & A:D:B:C \\ pA:D & + & B & \rightarrow & pA:D:B \\ pA:D & + & B:C & \rightarrow & pA:D:B:C \end{array}$$

Tabella 3.1: Lista delle reazioni implicate dalla freccia C_1 di Figura 3.1

Se si esamina nel dettaglio un simbolo di reazione in Figura 3.1, per esempio C_1, si possono elencare tutte le possibili reazioni che implicitamente rappresenta, come è stato fatto in Tabella 3.1.

Si noti che il numero di possibili reazioni cresce molto velocemente ogni volta che si aggiunge un singolo punto di interazione. Se per esempio si aggiungesse un legame non covalente tra B ed una nuova specie E, il numero delle reazioni che coinvolgono B raddoppierebbe visto che E potrebbe essere o non essere

legata. Al complicarsi della mappa l'operazione di enumerazione delle reazioni diventa presto un problema intrattabile, rendendo necessaria la ricerca di un metodo per fare a meno di doverle esplicitare.

In questa direzione si consideri che tutte le reazioni elencate interagiscono attraverso le porte P_3 e P_4; si può quindi affermare che la reazione C_1 coinvolge solo i complessi che hanno le porte P_3 e P_4 libere, oppure in altre parole, tutti i complessi che hanno le porte P_3 e P_4 libere, possono interagire utilizzando la reazione C_1.

Le considerazioni appena fatte sono di cruciale importanza. Infatti i reagenti per una particolare reazione verranno scelti in quell'insieme di complessi aventi le corrispondenti porte libere. Questa scelta non terrà conto di legami esistenti o di altri dettagli: l'unico fattore che candida un certo complesso ad essere scelto per una particolare reazione, è lo stato delle sue porte. Questa soluzione permette di descrivere i reagenti di ogni reazione in maniera implicita.

Definizione 3.2.1 *Una porta è il punto di contatto tra una interazione ed una specie molecolare.*

Tornando all'esempio: per C_1 i due insiemi di possibili reagenti corrispondenti a P_3 e P_4 sono mostrati in Tabella 3.2.

P_3	P_4
A	B
pA	$B : C$
$A : D$	
$pA : D$	

Tabella 3.2: Lista dei reagenti coinvolti dalle porte P_3 e P_4 di Figura 3.1.

Non è difficile notare che le parti sinistre delle reazioni di tabella 3.1, sono tutte e sole le combinazioni ottenibili partendo dai reagenti appena elencati. Questa lista di reagenti è completa ma ciò non significa che in ogni istante ognuno di essi sarà disponibile. All'inizio della simulazione verosimilmente saranno presenti solo specie molecolari elementari, quindi lentamente inizieranno a legarsi tra loro, dando forma a molecole non elementari. A causa delle diverse velocità di reazione, è plausibile considerare che non tutte le possibili combinazioni esistano in ogni momento.

Per questo motivo la simulazione delle mappe che verrà presentata nei prossimi paragrafi lascia che la rete evolva autonomamente, curandosi dei complessi man mano che appaiono. In questo modo si tiene traccia dei soli complessi che realmente esistono, del loro numero e di conseguenza di quali e quante porte sono libere. Come si vedrà, per ogni porta si terrà aggiornata in memoria una lista di tutti quei tipi di complessi che hanno quella porta disponibile.

3.3 Descrivere i complessi

Le reazioni che creano legami sono probabilmente le più utilizzate in questa notazione e creano un numero combinatoriamente enorme di possibili prodotti. Grazie all'uso che facciamo delle porte, un sistema semplice per descriverli è utilizzare i grafi.

Similmente a quanto accade nelle MIM, le specie molecolari vengono rappresentate come nodi, mentre le reazioni di complessazione come degli archi che partono e terminano in delle porte. Vediamo subito i dettagli ragionando sull'esempio di Figura 3.2. Il complesso $A : B$ ha due nodi A e B legati da un arco definito dalle porte P_3 e P_4. Visto che in una MIM un `port_id` identifica univocamente una porta e visto che ogni reazione utilizza una porta diversa, si può affermare che un arco è inequivocabilmente identificato da una coppia di porte. Così facendo conserviamo le informazioni relative ai legami interni al complesso esplicitamente nella sua descrizione.

Da adesso in poi, ogni nodo della MIM verrà chiamato `molecular_type` (per distinguerli dai nodi della rappresentazione interna dei complessi), identificato da un nome unico all'interno di ogni mappa: `molecular_type_id`.

In Figura 3.2 sono stati esplicitati i `molecular_type_id` delle varie specie molecolari elementari e non presenti: A, B e C vengono chiamate rispettivamente M_1, M_2 ed M_3; il nodo che rappresenta la specie non elementare $A : B$ ha `molecular_type_id` M_4 e serve principalmente a tener traccia delle possibili porte o inibizioni aggiunte come conseguenza della reazione su cui giace.

Inoltre ogni `molecular_type` ha un insieme finito di porte che li caratterizza: $M_1 : [P_1, P_2, P_3]$, $M_2 : [P_4]$, $M_3 : [P_6]$, $M_4 : [P_5]$. Si noti che il nodo di omodimerizzazione a sinistra nella mappa non ha un `molecular_type_id` tutto suo in quanto rappresenta semplicemente una seconda copia di A, difatti la porta P_1 appartiene al `molecular_type` M_1.

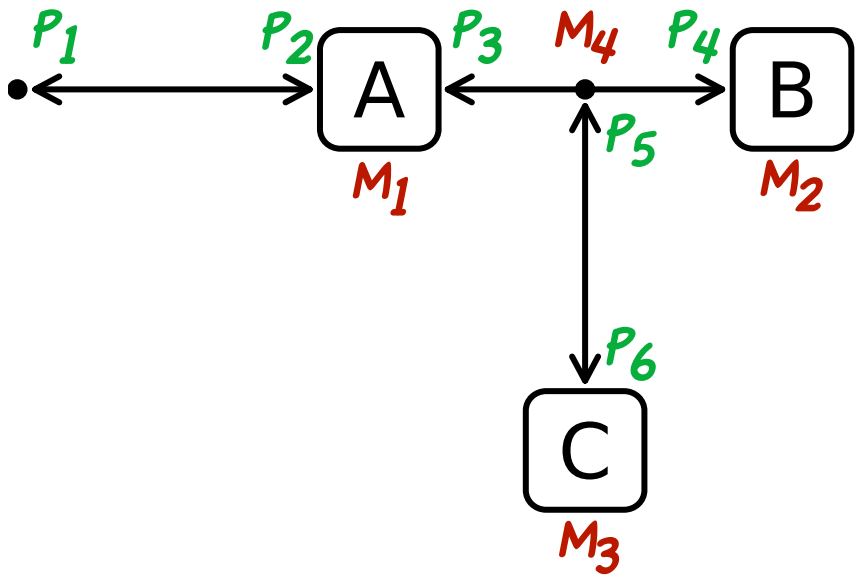


Figura 3.2: Una mappa nella quale sono stati assegnati dei nomi alle porte ($P_1, P_2, ...$) ed ai `molecular_type` (M_1, M_2, M_3 ed M_4).

Proprio a causa degli omodimeri serve un modo per distinguere due molecole di pari `molecular_type_id` all'interno di uno stesso complesso. Per questo motivo si introduce il `mol_id`, un valore intero che parte da 0, unico per ogni `molecular_type_id` in un complesso. Finalmente, la descrizione memorizzata per ogni nodo è una coppia (`molecular_type_id`, `mol_id`), mentre per ogni arco una coppia di coppie ((`mol_id`, `port_id`), (`mol_id`', `port_id`')). All'inizio di ogni simulazione esisteranno solo specie molecolari elementari, le cui descrizioni saranno le più semplici possibili: un singolo nodo con il corrispondente `molecular_type_id` e `mol_id` pari a 0 e nessun arco.

Per distinguere velocemente tra i vari complessi si utilizza un valore chiamato `complex_id`, unico per ogni tipo di complesso diverso. È importante sottolineare che questo valore si riferisce proprio al tipo di complesso, quindi alla sua struttura interna intesa come nodi ed archi. Ogni struttura diversa prenderà un nome diverso, come si capirà meglio nell'esempio del prossimo paragrafo.

Definizione 3.3.1 *Un `molecular_type` è un nodo della Mappa di Interazione Molecolare, definito dall'insieme delle sue porte e dalle contingenze che partono da esso.*

Definizione 3.3.2 *Un nodo nella descrizione di un complesso è una coppia (`molecular_type`, `mol_id`).*

Definizione 3.3.3 *Un arco nella descrizione di un complesso è una coppia di coppie* $((P_1, mol\_id_1), (P_2, mol\_id_2))$.

Definizione 3.3.4 *Siano* $C_1(N_1, E_1)$ *e* $C_2(N_2, E_2)$ *i grafi associati a due complessi e sia* $R(P_1, P_2)$ *una reazione di complessazione senza aggiunta di nodi che coinvolge le porte* P_1 *e* P_2. *Sia* M_1 *il* `mol_id` *della molecola che offre* P_1 *scelta all'interno di* C_1 *ed* M_2 *quello di* C_2 *che offre* P_2. *Siano* $\mathcal{R} : \{\mathbb{N}^+ \times 2^N\} \to 2^N$ *la funzione di riscrittura dei* `mol_id` *dove* 2^N *sono multiinsiemi di nodi, max il massimo valore* `mol_id` *di* C_1.

Allora il prodotto della reazione R *è il grafo definito dai nodi* $N_R = N_1 \cup \mathcal{R}(max, N_2)$ *e dagli archi* $E_R = E_1 \cup \mathcal{R}(max, E_2) \cup \{((P_1, M_1), (P_2, M_2 + max)\}$

3.3.1 Un esempio: descrizioni interne generate da tre reazioni

Questo esempio serve a vedere nel dettaglio le descrizioni interne che vengono generate per qualche semplice reazione. Considerando la rete di Figura 3.2, la simulazione inizia assegnando tre `complex_id` diversi alle tre molecole elementari, diciamo K_1 per la molecola di `molecular_type_id` pari a M_1, K_2 per M_2 e K_3 per M_3.

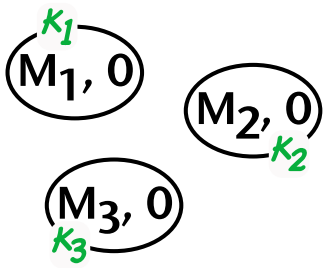


Figura 3.3: Rappresentazione interna di tre tipi di complessi (K_1, K_2 e K_3) con indicate le coordinate interne (`molecular_type`, `mol_id`).

La Figura 3.3 mostra questa situazione, in cui ovviamente i `mol_id` sono tutti pari a 0, non essendoci la necessità di distinguere nulla all'interno di complessi con un unico nodo.

Per ogni porta viene compilata una lista di `complex_id` aventi quella data

porta libera. L'unica lista vuota è quella di P_6 visto che non esiste ancora alcun complesso $A : B$. Le altre contengono un solo elemento: P_1, P_2 e P_3 vengono offerte solo da K_1, P_4 solo da K_2 ed infine P_6 da K_3.

Si supponga ora che il sistema abbia scelto la prima reazione e relativi reagenti (nei prossimi paragrafi si vedrà come questo accade) e che siano rispettivamente la complessazione $A : B$ (quindi attraverso le porte P_3 e P_4) ed i due complessi K_1 e K_2. La scelta dei reagenti implica che all'interno delle liste vengano individuate le porte coinvolte dalla reazione. Poichè al momento contengono un solo elemento la scelta diventa obbligata.

Essendo questa la prima reazione non esistono complessi $A : B$, quindi c'è bisogno di un nuovo `molecular_type_id` che verrà assegnato al prodotto della reazione, diciamo K_4. La descrizione del nuovo complesso includerà i due nodi (M_1, 0) e (M_2, 0) più M_4, il `molecular_type_id` che giace sull'arco di reazione utilizzato.

Si nota facilmente che i due nodi reagenti hanno gli stessi valori `mol_id`. Per mantenere la numerazione consistente all'interno del complesso che si sta creando, occorre riscrivere tali valori in almeno uno dei due reagenti. Questa operazione consiste nel prendere il più alto `mol_id` del primo reagente, diciamo n, per poi aggiungere $n + 1$ ai `mol_id` del secondo. Dopo aver effettuato questa rinumerazione è possibile unire gli insiemi dei nodi dei due reagenti ed aggiungere eventuali `molecular_type_id` che giacciono sull'arco, dopo aver assegnato loro un nuovo `mol_id`.

Nell'esempio, l'insieme di nodi risultante è $[(M_1, 0), (M_2, 1), (M_4, 2)]$.

Oltre ai nodi c'è bisogno di memorizzare anche gli archi. La reazione ne aggiunge uno tra le porte che ha utilizzato: ((0, P_3), (1, P_4)). In più bisogna connettere il `molecular_type` M_4 che giace sulla freccia di reazione. Questo viene fatto tramite un arco che lo collega al primo dei due reagenti, utilizzando due porte fittizie P_C, definite apposta per identificare tale situazione. La lista completa di archi del complesso appena formato è quindi $[((0, P_3), (1, P_4)), ((0, P_C), (2, P_C))]$. La situazione è visualizzata in Figura 3.4.

Arrivati a questo punto il sistema tiene traccia della diminuzione del numero di complessi K_1 e K_2 e della nascita dei K_4, che dispone anche della porta P_5 e verrà di conseguenza inserito nella lista corrispondente. Inoltre finirà anche nelle liste di P_1 e P_2 ma non in quelle di P_3 e P_4, visto che il complesso in

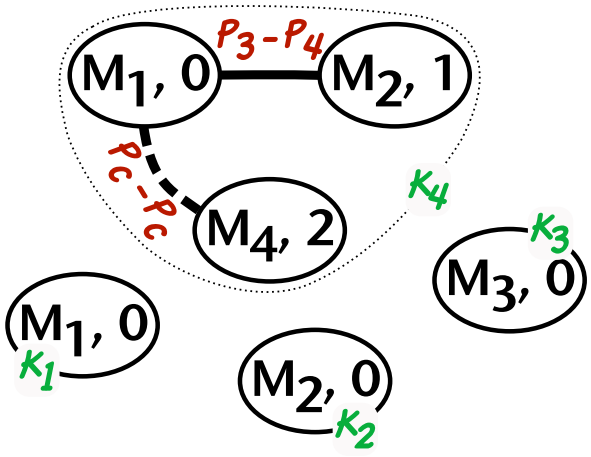


Figura 3.4: Rappresentazione interna: Oltre ai tre tipi di complessi elementari si è formato K_4, prodotto della reazione $A + B \rightarrow A : B$ di Figura 3.2. Sono riportati anche gli archi interni ai complessi $(P_3 - P_4, ...)$.

questione non le ha libere.

Si supponga ora che venga scelta la reazione di complessazione per creare $A : B : C$, quella tra P_5 e P_6. L'unico `complex_id` presente nella lista di P_5 è K_4, in quella di P_6 invece c'è solo K_3. I reagenti di questa reazione sono i `molecular_type` M_4 ed M_3; dopo la riscrittura dei `mol_id` viene quindi aggiunto il nodo $(M_3, 3)$, connesso tramite l'arco $((2, P_5), (3, P_6))$.

Visto che non esistono complessi di questa forma, c'è bisogno di un ulteriore `complex_id`, diciamo K_5. I suoi nodi sono:

$$[(M_1, 0), (M_2, 1), (M_4, 2), (M_3, 3)]$$

mentre gli archi:

$$[((0, P_3), (1, P_4)), ((0, P_C), (2, P_C)), ((2, P_5), (3, P_6))]$$

come mostrato graficamente in Figura 3.5.

A reazione avvenuta il sistema tiene traccia della scomparsa dei complessi K_4, dell'apparizione di K_5 e della diminuzione dei K_3. Aggiorna quindi di conseguenza le liste delle varie porte. Vale la pena di notare che la lista di P_5 viene svuotata: una volta eliminato K_4 la lista è vuota, il che significa che quella reazione non può più avvenire.

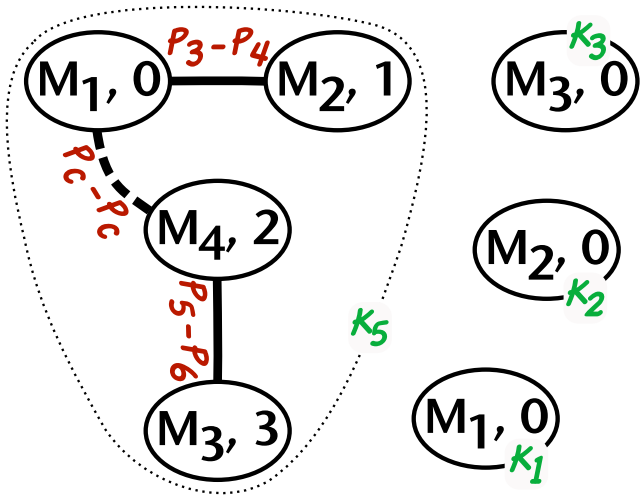


Figura 3.5: Rappresentazione interna: K_4 si è fuso con K_2 ed ha formato K_5, prodotto della reazione $A : B + C \rightarrow A : B : C$ di Figura 3.2. Sono riportati anche gli archi interni ai complessi $(P_3 - P_4, ...)$.

Ora si supponga che il sistema rifaccia la stessa sequenza di reazioni appena descritta e crei quindi la seconda copia del complesso K_5. Si supponga inoltre che venga scelta la reazione di omodimerizzazione tra le porte P_1 e P_2. Le liste in cui il sistema deve scegliere i complessi coinvolti contengono K_1 e K_5, supponiamo che scelga proprio le due istanze di K_5 e che inizi la complessazione partendo dalla riscrittura dei `mol_id`: il più alto valore in gioco è 3, per cui viene aggiunto 4 ai valori del secondo reagente. I due complessi vengono quindi uniti tramite un arco che connette due specie molecolari identiche di tipo M_1, che vengono distinte proprio grazie ai `mol_id` necessariamente diversi. Questo arco è $((0, P_1), (4, P_2))$ e visto che il complesso è nuovo il sistema crea un nuovo `complex_id`, diciamo K_6.

Il complesso K_6, visualizzato in Figura 3.6, ha quindi i seguenti nodi:

$$[(M_1, 0), (M_2, 1), (M_4, 2), (M_3, 3), (M_1, 4), (M_2, 5), (M_4, 6), (M_3, 7)]$$

ed archi:

$$[((0, P_3), (1, P_4)), ((0, P_C), (2, P_C)), ((2, P_5), (3, P_6)),$$
$$((4, P_3), (5, P_4)), ((4, P_C), (6, P_C)), ((6, P_5), (7, P_6)), ((0, P_1), (4, P_2))]$$

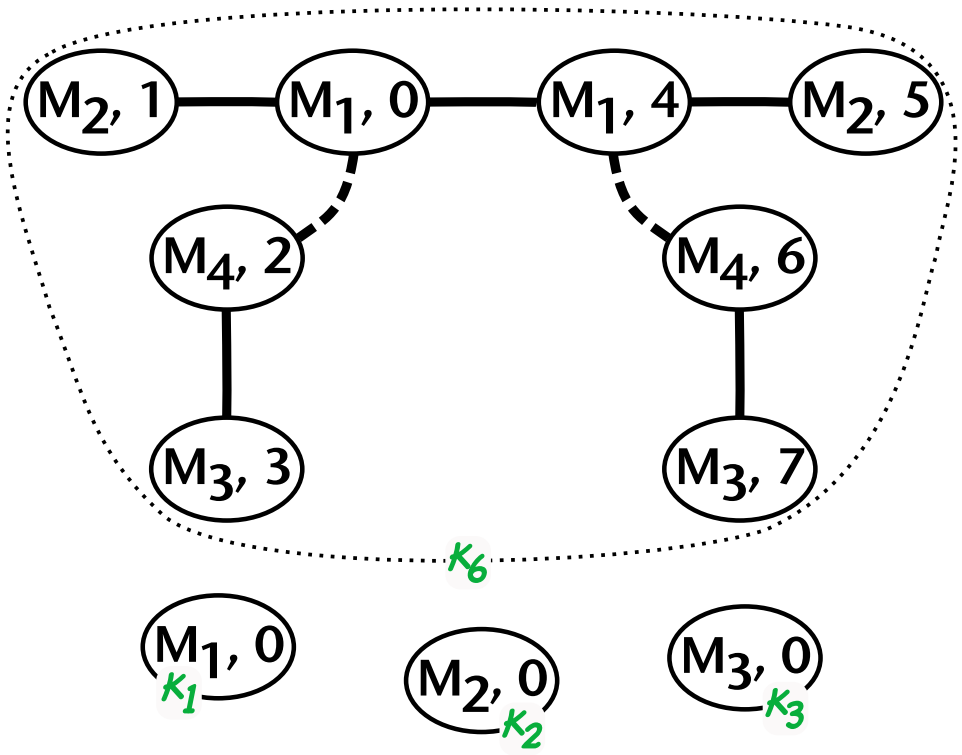


Figura 3.6: Rappresentazione interna: due copie del tipo molecolare K_5 si sono unite a formare K_6. La reazione è quella di omodimerizzazione di Figura 3.2.

Uno degli aspetti che rende questa codifica interessante è sicuramente rappresentato dalle *coordinate* che vengono assegnate ad ogni molecola all'interno di un complesso. In ogni istante il sistema sa quali sono le molecole che hanno delle porte libere, riuscendo ad eseguire le reazioni desiderate tra le giuste coppie di molecole. Altra cosa degna di nota è che grazie al `mol_id` gli omodimeri possono venir maneggiati senza paura. Come si vedrà nelle prossime sezioni, le liste mantenute per ogni porta non sono altro che una lista di coordinate nella forma (`complex_id`, `mol_id`).

3.4 Fosforilazione, rottura di legami covalenti e non covalenti

In Figura 3.7 si può vedere una mappa in cui le due molecole A e B possono legarsi tramite un legame non covalente a formare il complesso $A : B$. Inoltre

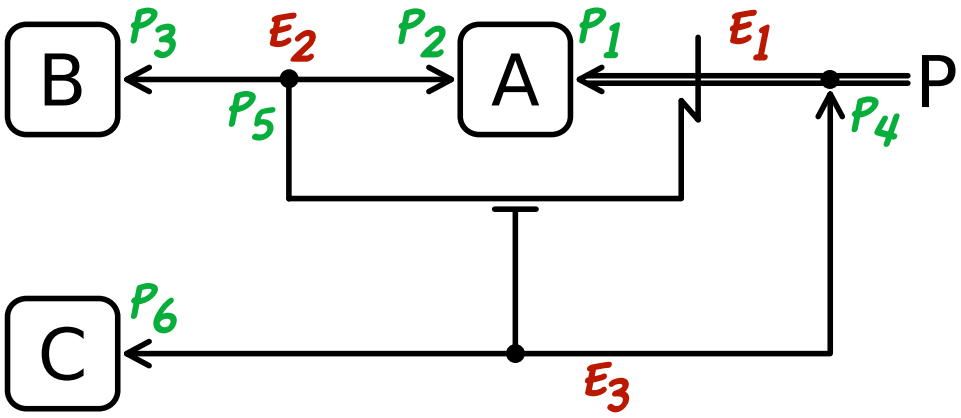


Figura 3.7: MIM in cui vengono assegnati dei nomi agli archi (E_1, E_2 ed E_3) oltre che alle porte (P_1, P_2, ...).

ad A si può attaccare un gruppo fosfato formando pA, il quale a sua volta può legarsi a C risultando in $pA : C$. Quando pA è legato a B invece, la fosforilazione può venir spezzata con una esplicita divisione del legame covalente.

Ulteriori reazioni possibili sono le naturali inversioni dei legami non covalenti tra A e B e tra pA e C, chiamate decomplessazioni. Queste reazioni hanno come obiettivo un arco: prendono la descrizione di un complesso e lo eliminano, dividendo spesso in due parti il complesso di partenza. La presenza ed il numero di tali archi all'interno del sistema simulato, viene tracciato in liste simili a quelle tenute per ogni porta. Per potersi riferire loro con facilità, la Figura 3.7 esplicita i loro nomi: $E_1 = (P_P, P_1)$ è l'arco che connette la particella di fosforilazione ad A[1], $E_2 = (P_2, P_3)$ collega A e B ed infine $E_3 = (P_4, P_6)$ connette pA con C. Infine i `molecular_type` sono M_1 per A, M_2 per B, M_3 per C, M_4 per il nodo prodotto dalla fosforilazione di A (il nodo con P_4 per intendersi) ed infine M_5 per il nodo che giace sulla complessazione tra A e B (quello a cui appartiene P_5).

La simulazione quindi inizia creando tre diversi `complex_id`: K_1 per A, K_2 per B e K_3 per C. Le porte libere sono P_1, P_2, P_3 e P_6, mentre P_4 e P_5 ancora non esistono. Nemmeno i tre archi appena descritti esistono, siamo nella situazione di partenza con sole specie elementari, in cui le reazioni possibili sono solamente la fosforilazione di A ed il legame non covalente tra A e B. Il sistema scelga proprio quest'ultima e, come visto nel paragrafo precedente, il

[1] P_P e' una porta speciale utilizzata per identificare la particella di fosforilazione.

complesso risultante, diciamo K_4, ha nodi $[(M_1, 0), (M_2, 1), (M_5, 2)]$ ed archi $[((0, P_2), (1, P_3)), ((0, P_C), (2, P_C))]$. I contatori vengono aggiornati diminuendo il numero dei reagenti ed aumentando il numero del prodotto K_4 che ora offre anche una porta P_4 ed un arco E_2. Questo significa che la reazione di decomplessazione $A : B \rightarrow A + B$ è ora possibile.

Si supponga che il sistema ora scelga la reazione di fosforilazione di K_4, formando K_5 di nodi:

$$[(M_1, 0), (M_2, 1), (M_5, 2), (P, 3), (M_4, 4)]$$

ed archi:

$$[((0, P_2), (1, P_3)), ((0, P_C), (2, P_C)), ((3, P_P), (0, P_1)), ((4, P_C), (0, P_C))]$$

come è illustrato in Figura 3.8

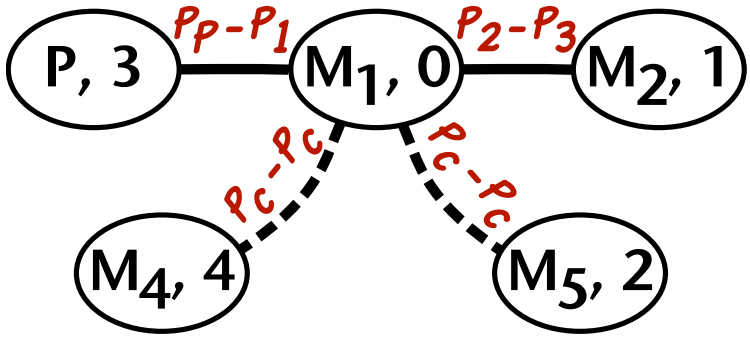


Figura 3.8: Rappresentazione del grafo associato a K_5, ricavato dalla MIM di Figura 3.7. Sono visualizzati i `molecular_type`, i `mol_id` ed i `port_id` che formano gli archi.

Il sistema ora sa che c'è una porta P_5 libera; gli archi E_1 ed E_2 sono disponibili rispettivamente per la divisione del legame covalente e per la decomplessazione, inoltre la reazione $pA : B + C \rightarrow pA : B : C$ è ora possibile.

Venga ora scelta la divisione del legame covalente. K_5 offre tutto ciò che serve: una porta P_5 libera ed un arco E_1. Il complesso risultante è esattamente K_4 e di conseguenza vengono aggiornati i contatori: non ci sono più archi E_1 ne porte P_5.

A questo punto venga scelta la decomplessazione: K_4 possiede l'arco E_2 che verrà spezzato. I complessi risultanti sono esattamente K_1 e K_2 ed i contatori

prendono gli stessi valori che avevano al primo passo di simulazione.

Molto similmente a come viene fatto per le porte, gli archi vengono tracciati tramite liste che permettono di sapere in ogni momento quali decomplessazioni e divisioni di legami covalenti sono possibili. Inoltre si noti anche qui l'importanza delle coordinate che permette di identificare efficacemente i giusti complessi coinvolti in tali tipi di reazioni.

3.5 Contingenze

Le contingenze sono simboli speciali, visto che influenzano sia le reazioni che altri simboli di contingenza, ma non le specie molecolari. In pratica modificano il comportamento di altri simboli, non aggiungono modi di interazione ma limitano quelli esistenti.

Dal momento che esistono delle regole di riscrittura che eliminano la stimolazione enzimatica e la stimolazione, le uniche contingenze di cui vale la pena preoccuparsi sono l'*inibizione* e la *necessità*, le cui interpretazioni sono esattamente speculari l'una all'altra. Se la reazione A *necessita* dell'entità B allora A dev'essere incapacitata fino all'apparizione di B. Viceversa se A viene *inibita* da B, la prima dev'essere incapacitata appena B appare.

In entrambi i casi, quello che accade in pratica, è l'impostazione di un semplice valore booleano connesso ad ogni istanza di ogni porta, che ci dice se è libera o meno. Questo valore booleano è gestito da speciali agenti di cui si parlerà in seguito.

Una regola di contingenza quindi non è altro che una implicazione della forma:

SE (un insieme di archi E è presente nel complesso)
ALLORA (alcuni archi e porte devono venir bloccate/sbloccate)

L'insieme di archi E che attiva la regola viene chiamato *testa* della contingenza: il complesso deve possederli tutti per poterla applicare. Se queste condizioni sono verificate allora bisogna applicare il *corpo* della contingenza, cioè bloccare o sbloccare le liste di porte ed archi specificate.

Le contingenze vengono descritte tramite liste per permettere di gestire le condizioni di AND logico descritte precedentemente: un singolo nodo che inibisce (o attiva) diverse entità oppure diversi nodi che tutti assieme inibiscono (o attivano) una certa reazione.

Il luogo più adatto per memorizzare queste contingenze, come si vedrà, è nelle descrizioni dei `molecular_type` da cui partono i simboli che le rappresentano. In questo modo appena viene aggiunto un nodo ad un complesso basta scorrere la lista delle sue contingenze ed applicare quelle che vengono verificate dalla struttura del complesso stesso.

3.6 L'algoritmo di simulazione

Dopo aver afferrato il principio che sta dietro alla codifica in sCCP (e di conseguenza Prolog) di questa notazione, è necessario introdurre il modo in cui verrà gestita la simulazione.

La gestione delle competizioni stocastiche di sCCP è ottenuta grazie ad un adattamento del celebre algoritmo di Gillespie ([12], [13]), già utilizzato da anni nelle simulazioni di reazioni chimiche. Per la precisione, come visto nei precedenti paragrafi, viene generata una catena di Markov della quale viene calcolata una possibile traccia grazie all'algoritmo di Gillespie. Questo assicura che la scelta del prossimo agente da eseguire rispetterà le proprietà esposte nei precedenti capitoli.

Gli agenti hanno bisogno inoltre di un meccanismo di sincronizzazione per comunicare, ottenuto grazie alle guardie tramite `ask` bloccanti.

Dopo una prima fase di inizializzazione del sistema, il motore di simulazione delle MIM si assesterà sul seguente ciclo di 4 passi:

1. scegli la prossima reazione R
2. scegli i reagenti di R
3. crea il prodotto P
4. applica le contingenze di P

Essi sono realizzati, come detto, in parte dal metainterprete ed in parte negli agenti stessi. La scelta della reazione si basa pesantemente sui predicati di conteggio delle varie quantità, dunque sulle funzioni che calcolano i rate di cui si è accennato nel precedente paragrafo. Il modello stocastico sottostante, infatti, segue il principio di *mass action*, secondo il quale la velocità di ogni reazione è proporzionale alla quantità di ognuno dei reagenti.

Si capisce l'importanza dell'utilizzo delle porte: una reazione R, definita in termini delle porte P_1 e P_2, è possibile se e solo se esiste almeno una istanza per ognuna delle sue porte. Inoltre, la probabilità che sia proprio R ad essere scelta è direttamente proporzionale al numero di P_1 e di P_2. Questo comportamento viene ottenuto tramite una competizione tra agenti, un race stocastico, i quali sono attivi se la reazione a cui fanno capo è possibile, in base al numero delle sue porte appunto.

Scelta la reazione la strada è in discesa. L'agente di reazione vincitore attiva quello che gestisce le porte di cui ha bisogno, il quale pesca in modo casuale i complessi che verranno coinvolti come reagenti, comunicandoglieli. Dunque si costruisce il prodotto ed infine si scorre la lista delle contingenze ad esso collegate, applicando quelle possibili. Una volta aggiornati i contatori di porte, archi, complessi reagenti e prodotto si può ripartire da capo.

Capitolo 4

Codice sCCP

In questo capitolo viene presentato e discusso il codice sorgente del simulatore. Nella prima parte vengono analizzati i predicati utilizzati a run time mentre nella seconda si prendono in esame gli agenti con i relativi predicati.

Durante la trattazione verranno omessi predicati di importanza minore e trascurati particolari riguardo al linguaggio, per alleggerire l'esposizione. Tuttavia questi particolari sono necessari per poter eseguire la simulazione, perciò chi volesse approfondirli può leggere i file sorgenti dei due interpreti sCCP e delle Mappe di Interazione Molecolare facendo riferimento a [4] per la semantica e la sintassi.

Nei paragrafi che seguono verranno introdotti i predicati che il sistema maneggia a tempo di esecuzione e gli agenti che lo fanno, presentandone la struttura ed il funzionamento oltre ai modi in cui interagiscono tra loro.

4.1 I predicati

Qui di seguito viene presentata la struttura di predicati che il sistema gestisce a tempo di esecuzione. Probabilmente il più importante è `complex`, che contiene le descrizioni delle molecole elementari e non, presenti nel sistema in ogni momento. Esso si basa sulla presenza del predicato `molecular_type`, usato per descrivere le proprietà dei nodi della MIM. Altro predicato importante è `contingency` contenente appunto le descrizioni delle contingenze nel formato descritto in precedenza. Altri predicati vengono usati per registrare quali e quanti archi e porte sono libere in un dato momento.

4.1.1 Predicati per definire i dati di ingresso del simulatore

Per definire i dati da passare come ingresso al simulatore è necessario istanziare alcuni predicati che definiscono principalmente le molecole elementari in gioco, il loro numero e le contingenze che implicano.

Il predicato che definisce le specie elementari è `molecular_type`, la cui struttura è definita nella Tabella 4.1. Il compito di questo predicato è quello di definire l'insieme di porte e contingenze di ogni nodo della MIM, istanziando un predicato di questo tipo per ogni specie elementare della mappa. I nomi qui definiti (`molecular_type_id`) vengono usati nei predicati che descrivono i complessi e sono fondamentali quando si debba risalire alla lista di contingenze da verificare per un dato complesso. Inoltre deve venir istanziato uno di questi predicati per ognuno dei nodi creati come conseguenza dell'interazione tra specie elementari (come M_4 in Figura 3.3).

`molecular_type`(`molecular_type_id`, `port_list`, `contingency_list`)

`molecular_type_id`	Nome (unico all'interno di una mappa) di questa specie elementare.
`port_list`	Lista di identificatori di porte ([`port_id`, `port_id`', ...]) appartenenti a questo complesso. Ogni porta ha un identificatore unico all'interno di una mappa ed appartiene ad un solo `molecular_type`. Non sono utilizzabili i nomi `pc` e `pp`, riservati dal simulatore per descrivere rispettivamente legami con `molecular_type` fittizi e particelle di fosforilazione.
`contingency_list`	Lista di identificatori di contingenze ([`contingency_id`, `contingency_id`', ...]) che fanno capo a questo nodo nella MIM.

Tabella 4.1: La struttura del predicato `molecular_type`.

Il predicato che memorizza le contingenze (Tabella 4.2), come detto, è piuttosto semplice. Contiene due liste che indicano quali sono le entità che disattiva (o attiva) e chi viene disattivato (o attivato) dalla contingenza in questione. Un campo speciale permette di distinguere tra inibizioni e necessità

e dunque determina il comportamento che il gestore della simulazione deve tenere quando essa viene verificata.

`contingency(contingency_id, inh, head, tail)`

`contingency_id`	Nome (unico all'interno di una mappa) che identifica univocamente questa contingenza.
`inh`	Valore booleano pari a *true* se questa contingenza è una inibizione, *false* se è una necessità.
`head`	Lista di archi nel formato (`port_id1`, `port_id2`) che il complesso deve possedere per poter dichiarare verificata questa contingenza.
`tail`	Coppia di liste (`ports`, `edges`) dove `ports` è una lista di `port_id` ed `edges` è una lista di archi nel formato (`port_id1`, `port_id2`); entrambe le liste specificano gli oggetti che verranno inibiti/attivati se la contingenza è verificata.

Tabella 4.2: La struttura del predicato `contingency`.

Inoltre è necessario istanziare un predicato `target_edges(edge_list)` contentente una semplice lista di archi che saranno oggetto di decomplessazione o divisione di legame covalente. La lista è quindi del tipo [(`P1`, `P2`), ...], una coppia per ogni complessazione ed ogni divisione presente nella mappa.

L'ultimo predicato da definire per completare i dati di ingresso del simulatore riguarda le quantità delle specie elementari e si chiama `elementary_species`. L'unico contenuto è una lista di coppie tipo

$$[(\texttt{molecular\_type}, \texttt{n}), (\texttt{molecular\_type}', \texttt{n}'), ...]$$

una coppia per ogni `molecular_type` definito in precedenza. `n` definisce la quantità iniziale che quella specie elementare avrà all'interno del sistema che si vuole simulare. Per i `molecular_type` aggiunti come conseguenza di una interazione va specificata una quantità pari a -1. Questo perchè il simulatore automaticamente crea, per ognuno di questi predicati, un banale `complex_type` con un solo nodo e li rende subito disponibili per essere coinvolti dalle reazioni. Per quei `molecular_type` la cui quantità è pari a -1 questo non accade.

4.1.2 Il predicato `complex`

Il predicato `complex`, mostrato in Tabella 4.3, essenzialmente descrive i grafi associati alle strutture effettivamente presenti nel sistema a tempo di esecuzione. Il motore di simulazione deve quindi poterlo modificare, aggiungendo e rimuovendo nodi ed archi a seconda delle occorrenze. Ci sono alcuni casi in cui si usano dei valori particolari:

- fosforilazione: viene aggiunto un nodo speciale con `mol_type_id` $= P$ connesso da un arco con entrambi i `port_id` $= P_P$
- quando una reazione aggiunge al complesso punti di interazione, cioè porte, oltre a quelle dei reagenti, viene aggiunto il nodo speciale che li contiene (definito dalla reazione che si sta eseguendo) tramite un arco fittizio con entrambi i `port_id` $= P_C$ e connesso ad uno dei due reagenti.

`complex(complex_type, nodes, edges)`

`complex_type_id`	Identifica univocamente ogni tipo di complesso generato a tempo di esecuzione.
`nodes`	Lista di nodi nel formato (`mol_type_id`, `mol_id`) in cui `mol_id` è unico all'interno di ogni tipo di complesso (vengono eseguiti controlli e re-numerazioni ad ogni reazione di complessazione/decomplessazione) e viene usato per distinguere molecole elementari dello stesso tipo all'interno di un tipo di complesso.
`edges`	Lista di archi del complesso nel formato ((`mol_id1`, `port_id1`), (`mol_id2`, `port_id2`)); i valori `mol_id` sono consistenti a quelli del campo `nodes`.

Tabella 4.3: La struttura del predicato `complex`.

Si noti che questo predicato è coinvolto dalle operazioni più importanti e delicate che il simulatore svolge. Oltre a descrivere quanto esiste a tempo di esecuzione, viene utilizzato per riconoscere se un dato complesso esiste già oppure no. Questa problematica, come si vedrà nei prossimi paragrafi, è decisamente delicata: come già argomentato le mappe permettono di creare uno stesso prodotto in più modi. Ciò significa che uno stesso complesso potrebbe apparire con due descrizioni sintatticamente diverse. Si capisce quindi come

questo predicato debba essere il più preciso e rigido possibile ma allo stesso tempo flessibile, per poter essere maneggiato con facilità.

4.1.3 Predicati di conteggio

Fondamentali ai fini di una corretta esecuzione, i predicati che tengono nota delle quantità delle varie entità in gioco sono molteplici. Oltre a conoscere esattamente il numero di copie delle varie specie elementari, infatti, c'è bisogno di sapere quante porte libere di ogni tipo sono presenti nel sistema, quanti archi e, soprattutto, bisogna collegare tra loro queste informazioni, per poter risalire velocemente all'intera descrizione di un complesso a partire da una semplice porta.

Il predicato `complex_type_num(complex_type_id, n)` lega il numero di istanze presenti nel sistema simulato ad ogni `complex_type`. In maniera del tutto simile `port_num(port_id, n)` ed `edge_num((port_id1, port_id2), n)` contano quante porte di quel dato id e quanti archi di quel tipo esistono in ogni dato momento.

Infine i predicati che collegano tra loro queste informazioni sono per le porte `port_complexes` e per gli archi `edge_complexes`; la struttura del primo è specificata in Tabella 4.4. Il secondo è molto simile: una delle differenze risiede nelle coordinate necessarie per identificare univocamente l'arco di cui si sta tenendo conto. Infatti per puntare un arco non basta `mol_id` ma ci vuole una coppia di coppie ((`mol_id1`, `port_id1`), (`mol_id2`, `port_id2`)). Quindi, in questo caso, il predicato `complex_list` memorizza una lista di elementi del tipo (`complex_type_id`, ((`mol_id1`, `port_id1`), (`mol_id2`, `port_id2`)), `inh`). L'altra differenza è nel primo campo, in un caso `port_id` e nell'altro la coppia (`P1`, `P2`), che identifica un arco invece che una porta.

4.2 Gli agenti

Gli agenti sono sicuramente il cuore dell'apparato simulatore, per questo motivo vale la pena spendere qualche parola per giustificarne la struttura ed il comportamento. Volendo ricapitolare i principali concetti alla base della presente codifica:

- ogni nodo è identificato dall'insieme delle proprie porte
- le reazioni vengono modellate come agenti, istanziati a seconda delle esigenze

`port_complexes(port_id, comples_list)`

`port_id`	Porta a cui fa riferimento questo predicato.
`complex_list`	Lista di triple della forma (`complex_type_id`, `mol_id`, `inh`); i primi due sono le coordinate che permettono di identificare con precisione il complesso e la giusta molecola al suo interno, `inh` è un valore booleano che se uguale a *true* significa che la porta in questione è inibita, se *false* significa che è libera.

Tabella 4.4: La struttura del predicato `port_complexes`.

- un complesso viene memorizzato come un grafo; l'insieme di porte ad esso associato è determinato dall'insieme dei nodi
- i reagenti coinvolti da una reazione vengono scelti all'interno dell'insieme dei complessi aventi quelle determinate porte libere

Per questi motivi si è deciso di creare delle entità che si occupassero delle liste associate ad ogni porta.

4.2.1 Gestione delle porte e degli archi

Il primo passo dell'algoritmo di simulazione prevede la scelta di reagenti che, come detto, avviene tramite un race stocastico fra gli agenti reazione attivi, cioè quelli che hanno un numero maggiore di zero di porte associate libere. L'agente che vince questa gara sarà l'unico che potrà proseguire la propria esecuzione.

Questa scelta viene delegata a delle entità, chiamate *port manager* (ed *edge manager*), il cui compito è rimanere in attesa che un agente reazione li interpelli, effettuare la scelta all'interno dell'apposita lista, comunicarla per poi tornare a dormire. Nel Listato 4.1 si vede il codice sCCP di uno di questi agenti: la prima istruzione è un `ask` bloccante di sincronizzazione: l'agente reazione interessato a risvegliare il port manager deve modificare il predicato `port_man_status(Port_id, Active, Done)` nel constraint store[1],

[1] è necessario eliminarlo dal constraint store con l'istruzione `retract` ed aggiungerlo con i nuovi valori ramite `assert`

impostando `Active` ad 1. Dopo esser stato svegliato viene eseguita la scelta tramite il predicato `choose_complex(Port_id, Complex_id, Mol_id))`, il quale restituisce le due coordinate necessarie ad identificare la molecola all'interno del complesso. Queste coordinate vengono memorizzate nel constraint store in un apposito predicato (`chosen_coordinates(Port_id, Complex_id, Mol_id)`) ed infine, impostando `Done` a 1 nel predicato `port_man_status`, si comunica all'agente reazione la fine della procedura di scelta.

Listato 4.1: L'agente `port_manager`

```
1 port_manager(Port_id) :->
2  ask(is_port_manager_active(Port_id))@rate(fast ,[])
3  : tell (choose_complex(Port_id, Complex_id, Mol_id))@rate(fast,[])
4  : tell (complex_chosen(Port_id, Complex_id, Mol_id))@rate(fast,[])
5  : tell (port_manager_done(Port_id))@rate(fast,[])
6  : port_manager(Port_id)@rate(fast,[]) .
```

La scelta di un arco da parte di un edge manager avviene esattamente allo stesso modo, la differenza risiede ancora una volta nelle coordinate comunicate all'agente reazione, in grado di identificare un arco come coppia di coppie `(Complex_id, Mol_id)`.

Listato 4.2: L'agente `port_man_spawner`

```
1 port_man_spawner :->
2  ask(is_spawn_list_not_empty)@rate(fast, [])
3  : tell (get_next_port(Port))@rate(fast, [])
4  : tell (init_port_manager(Port))@rate(fast, [])
5  : ( port_manager(Port)@rate(fast, []) & port_man_spawner@rate(fast, []))
6  ++ ask(is_spawn_list_empty)@rate(fast, []) .
```

Questi gestori di porte ed archi devono venir lanciati all'inizio dell'esecuzione del programma sCCP, chiamando `port_man_spawner` (ed il simmetrico per gli archi `edge_man_spawner`), di cui si può vedere il codice sorgente nel Listato 4.2. Questi speciali agenti hanno il solo compito di scorrere la lista di porte definita dalla mappa (tramite il predicato `get_next_port`) e lanciare un `port_manager` per ognuna di esse. Tramite una scelta stocastica con guardie mutualmente esclusive si controlla se la lista non è ancora finita (`is_spawn_list_not_empty`) caso in cui ricorsivamente richiama se stesso, altrimenti semplicemente termina.

4.2.2 Calcolo dei rate

Come detto, il calcolo dei rate avviene secondo il principio di *mass-action*, per il quale la velocità di ogni reazione è proporzionale al numero dei reagenti presente nel sistema. Da questo punto di vista si distinguono due famiglie di reazioni: quelle con un unico reagente e quelle con due. A quest'ultima appartiene il legame non covalente, la complessazione, il cui rate è proporzionale al prodotto del numero di entrambi i reagenti. Nel Listato 5.2 si può vedere il predicato che lo calcola, dopo aver estratto dal constraint store le quantità esegue la moltiplicazione e restituisce il risultato. Il nome di tale predicato viene passato come parametro dall'agente sCCP ed il metainterprete di simulazione lo eseguirà, come descritto nei precedenti paragrafi.

Listato 4.3: Il predicato per calcolare il rate delle complessazioni

```
1 comp_r(Out_rate, Port1, Port2, Rate) :-
2   port_num(Port1, N1),
3   port_num(Port2, N2),
4   Out_rate is N1*N2*Rate.
```

Il calcolo dei rate per l'altra famiglia di reazioni è addirittura più semplice: dopo aver ricavato il numero dell'unico reagente lo si moltiplica per il rate base e si restituisce il risultato.

4.2.3 Gli agenti di reazione

Come anticipato nei precedenti paragrafi, ogni agente di reazione ha una istruzione di `ask` bloccante, il cui rate definisce la probabilità che venga eseguito e la sua velocità. Per non sovrastimare la durata della reazione, le successive istruzioni sCCP dell'agente dovrebbero essere istantanee. Tuttavia, per preservare le proprietà riguardanti le catene di Markov, il linguaggio non offre una istruzione di `ask` instantanea.

L'espediente utilizzato per ovviare a questo ostacolo è associare un rate *fast* talmente grande che la probabilità che il linguaggio scelga proprio questa istruzione invece di una con rate normale è praticamente uno. Inoltre la sua durata temporale risulta talmente piccola da non risultare significativa se confrontata con la durata tipica di una reazione. Il risultato, a tutti gli effetti, è un `ask` istantaneo, che non consuma tempo della simulazione e viene eseguito sicuramente senza attese.

L'agente che esce vincitore dalla competizione stocastica, deve per prima cosa disabilitare tutti gli altri. Questo meccanismo è ottenuto tramite due predicati: `lock_reactions`, che blocca le reazioni, ed `are_reactions_unlocked`, che entra in gioco all'interno dei predicati guardia (come `complexation_guard`). Quindi il primo agente che supera il proprio `ask` bloccante automaticamente blocca tutti gli altri.

A questo punto l'agente si sincronizza con il proprio `port_manager` (oppure con `edge_manager` nei casi di decomplessazione e divisione esplicita). Nel caso di `complexation_mm` del Listato 5.1, questa operazione viene svolta sequenzialmente per due volte, per ottenere appunto i due reagenti da legare. Dopo averli ottenuti, il predicato `build_complex` si occupa di unire i loro grafi aggiungendo il nuovo arco ed il `molecular_type` specificato nel parametro `middle_mol_type`.

È ora possibile memorizzare la descrizione in memoria, tramite `add_complex`, il quale decide se questo grafo esiste già o meno (nel prossimo paragrafo si vedrà come questo accade) e di conseguenza gli assegna il `complex_id` trovato o ne crea uno nuovo.

Listato 4.4: L'agente `complexation_mm`

```
1  complexation_mm(Port_id1, Port_id2, Middle_mol_type, Rate) :->
2   ask(complexation_guard(Port_id1, Port_id2))@rate(comp_r, [Port_id1, Port_id2,
        Rate])
3   : tell ( lock_reactions )@rate(fast ,[])
4   : tell (activate_port_manager(Port_id1))@rate(fast ,[])
5   : ask(is_port_manager_done(Port_id1))@rate(fast ,[])
6   : tell (activate_port_manager(Port_id2))@rate(fast ,[])
7   : ask(is_port_manager_done(Port_id2))@rate(fast ,[])
8   : tell (get_chosen_complex(Port_id1, Complex_id1, Mol_id1))@rate(fast,[])
9   : tell (get_chosen_complex(Port_id2, Complex_id2, Mol_id2))@rate(fast,[])
10  : tell (build_complex(Complex_id1, Complex_id2, Port_id1, Port_id2, Mol_id1,
        Mol_id2, Middle_mol_type, Edges, Nodes, Contingencies))@rate(fast,[])
11  : tell (add_complex(Edges, Nodes, Contingencies, New_complex_id))@rate(fast
        ,[])
12  : tell (decrease_numbers_of_complex(Complex_id1))@rate(fast,[])
13  : tell (decrease_numbers_of_complex(Complex_id2))@rate(fast,[])
14  : tell (increase_numbers_of_complex(New_complex_id))@rate(fast,[])
15  : tell (apply_contingencies(New_complex_id))@rate(fast,[])
16  : tell (reset_port_manager(Port_id1))@rate(fast ,[])
17  : tell (reset_port_manager(Port_id2))@rate(fast ,[])
18  : tell (unlock_reactions)@rate(fast ,[])
19  : complexation(Port_id1, Port_id2, Rate)@rate(fast ,[]) .
```

I due `decrease_numbers_of_complex` e `increase_numbers_of_complex` gestiscono le liste delle porte libere, quelle degli archi ed i predicati di conteggio dei complessi. In questo caso per esempio, bisogna sottrarre uno al numero dei due reagenti e delle due porte utilizzate ed aggiungere uno al numero di complessi prodotto, al numero dell'arco appena creato e delle eventuali nuove porte sbloccate dal processo.

Infine `apply_contingencies` scorre la lista delle contingenze associate a tutti i `molecular_type` appartenenti al grafo verificando se le condizioni ivi definite sono verificate.

Si noti che grazie al modo in cui è definita la procedura di complessazione, questa operazione è necessaria solo alla prima apparizione di un complesso. Infatti se è stato decretato che esiste già un grafo di questa forma, significa che in passato sono già state applicate le stesse identiche contingenze, le quali hanno semplicemente modificato le liste relative ai gestori di porte ed archi. Queste liste fanno riferimento ad un tipo di complesso e non a questa particolare nuova istanza. In pratica, in questo caso, applicare le contingenze significherebbe dover riscrivere gli stessi valori nelle stesse posizioni delle stesse liste, per tale motivo è sufficiente incrementare i contatori.

Oltre a quella appena presentata, esiste una seconda versione dell'agente di complessazione la quale però non aggiunge alcun nodo aggiuntivo, quindi manca del parametro `middle_mol_type`. Questa diversificazione vale anche per gli agenti di decomplessazione, e si possono distinguere grazie al suffisso `_mm`.

4.2.4 Riconoscere i complessi

I predicati logici come `add_complex` nascondono uno dei problemi più difficili della codifica: esiste già in memoria un complesso della stessa forma? La questione è delicata, visti i molteplici modi esistenti per creare uno stesso complesso. Per esempio semplicemente invertendo l'ordine dei due reagenti si otterrebbero, in linea di principio, due rappresentazioni dei complessi sintatticamente diverse.

Un primo passo verso la soluzione, è introdurre un semplice ordinamento per i `molecular_type`, per le porte e gli archi. Rispettando questi ordinamenti, cioè scambiando all'occorrenza primo e secondo reagente, durante le tutte le operazioni che modificano i grafi (legami covalenti e non) si ha la sicurezza che essi vengono creati sempre allo stesso modo, rendendo quindi possibile un semplice confronto alfabetico fra le liste salvate in memoria.

Purtroppo per un complesso passato attraverso una reazione di omodimerizzazione[2] questi accorgimenti non servono più. È facile costruire un esempio in cui non è possibile definire uno o più ordinamenti in grado di assicurare consistenza tra le descrizioni ottenute tramite diverse sequenze di reazioni.

Una soluzione che funziona in qualsiasi caso è quella di considerare il problema come isomorfismo di grafi, problema di non facile soluzione. Tuttavia, come già accaduto in diverse occasioni ([25], [26]) è possibile sfruttare le caratteristiche della notazione per ridurre il problema ad un caso più semplice, per cui è possibile costruire un algoritmo di complessità polinomiale.

Nello specifico è possibile utilizzare un algoritmo di visita che lavori in parallelo su due descrizioni di complessi. La prima operazione da fare è scegliere un nodo di partenza. Nel caso medio si possono sfruttare gli ordinamenti definiti su nodi ed archi e scegliere un tipo molecolare che appare una volta sola nella descrizione dei complessi. A quel punto l'algoritmo di visita procede controllando gli archi uscenti e ricorsivamente sui nodi a lui connessi, arrestandosi appena trova una incongruenza o al completamento della visita. Grazie alle restrizioni imposte sull'unicità delle porte ed alla proprietà per la quale esiste una unica coppia di porte che identifica un arco, l'algoritmo descritto risolve il problema di riconoscere se due complessi sono o non sono uguali.

Il caso pessimo è rappresentato da un complesso in cui risulta difficile identificare un nodo di partenza, per esempio il prodotto di omodimerizzazioni e complessazioni, in cui esistono n copie di un certo `molecular_type` ed m di un altro. In questo caso non ci sono tipi molecolari che appaiono una sola volta per cui obbligatoriamente bisogna far partire più volte la visita da ognuno dei nodi di un certo tipo, risultando in una complessità al più quadratica, $|N|$ volte la complessità della visita.

Per fortuna la maggior parte dei casi è risolvibile tramite il confronto tra gli archi dei due complessi, direttamente tramite l'algoritmo di unificazione di Prolog: se non sono esattamente gli stessi si può affermare che sono diversi. Chiaramente questo riduce notevolmente il numero dei casi in cui occorre visitare il grafo associato.

Lo stesso problema si presenta in fase di rottura di un legame covalente o

[2] cioè due istanze della stessa specie elementare A si legano tra loro creando un complesso tipo $A : A$

non. Eliminando un arco da un grafo ci sono due possibilità: ottenere comunque un grafo connesso oppure ottenerne due. Dopo aver scoperto in quale caso ci si trova tramite una semplice visita, bisogna risolvere il problema di isomorfismo con i grafi presenti nel constraint store, per risalire agli eventuali `complex_id` o per crearne di nuovi.

Capitolo 5

Simulare una rete biologica

Per testare la codifica delle Mappe di Interazione Molecolare ottenuta, si è scelto di partire da un sistema semplice: una semplificazione della transizione da G1 ad S del ciclo cellulare dei mammiferi. La mappa è stata presa da un articolo di Kohn stesso [19], il quale parte da una rete molto semplice e via via la raffina per ottenere gradualmente un modello verosimile. Questo approccio evolutivo permette di isolare i comportamenti ed i meccanismi di diverse entità in gioco, per poi assemblarle sfruttando la composizionalità offerta dalla notazione.

La presentazione che seguirà ometterà alcuni particolari relativi ad aspetti poco importanti relativi principalmente al linguaggio utilizzato. Lo scopo è quello di presentare il codice sorgente sottolineando i punti di forza ed evidenziando quelli deboli, senza appesantire inutilmente la discussione. Di conseguenza il codice presentato non è direttamente eseguibile, o meglio interpretabile; chi volesse venire a conoscenza anche dei particolari qui omessi, può consultare i file sorgenti dei due interpreti e dell'esempio che verrà presentato.

5.1 Un esempio: la transizione dalla fase G1 alla S del ciclo cellulare

La transizione nella fase S del ciclo cellulare è regolata da una rete di geni in cui sono compresi i fattori *Rb* ed *E2F*. Il primo, acronimo di retinoblastoma, legandosi al secondo blocca l'ingresso della cellula nella fase S[1]. Inoltre E2F viene sintetizzato ad un ritmo costante e degradato da una proteasi.

[1] in verità anche Rb è regolata tramite fosforilazione, ma in questa mappa tale meccanismo è stato semplificato omettendolo.

Il fenomeno che questa rete si prefigge di studiare è un ritardo nell'accumulo

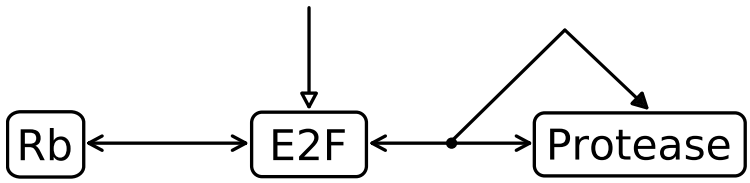


Figura 5.1: Semplificazione di una parte di rete biologica che descrive la transizione dalla fase G1 alla S.

del fattore $E2F$, ad opera appunto della proteina Rb. Infatti, se è presente, si lega ad $E2F$, inabilitando la degradazione ad opera della proteasi. Appena gli Rb sono stati saturati, la concentrazione di $E2F$ sale fino ad uno stato di equilibrio, nel quale rimane costante. Dunque la presenza della proteina Rb ritarda il raggiungimento di questo equilibrio.
Questo semplice modello suggerisce come un semplice fattore Rb possa controllare il ritardo nella duplicazione del DNA tramite la regolazione della proteina $E2F$.

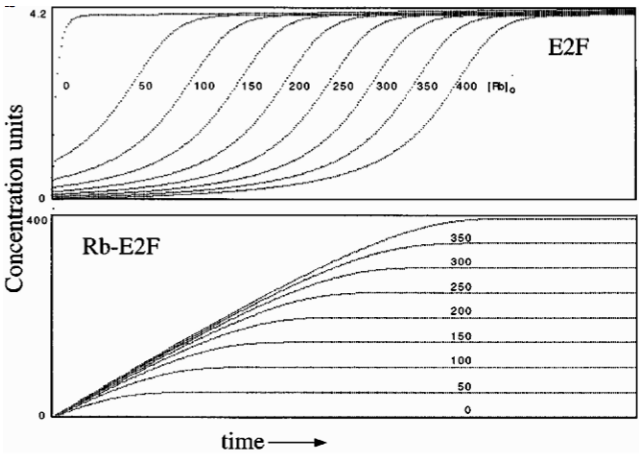


Figura 5.2: Grafici tratti dall'articolo originale da cui è stato preso l'esempio ([19]). Le varie linee rappresentano diverse concentrazioni iniziali della proteina Rb.

La mappa di Figura 5.1 racchiude proprio questo meccanismo. $E2F$ viene sintetizzata ad un rate costante, si può legare ad Rb, oppure a protease. Quest'ultimo complesso può venir convertito nuovamente in protease, modellando

il meccanismo di degradazione.

Nell'articolo da cui è stato tratto questo esempio [19], la simulazione avviene tramite un approccio basato su equazioni differenziali ordinarie (ODE), interpretando la mappa in modo esplicito. Il risultato ottenuto è visibile in Figura 5.2: si nota un accumulo di complessi $Rb : E2F$ che si arresta una volta saturata la proteina Rb, contemporaneamente al diminuire di Rb disponibili c'è un accumulo ritardato di proteine $E2F$ libere. I grafici sono ottenuti per diverse valori iniziali di Rb: risulta evidente che il ritardo subito dal sistema è tanto maggiore quanto è più alta la concentrazione iniziale di Rb.

5.2 Manipolazione iniziale della mappa

La primissima operazione da fare per cercare di ottenere gli stessi risultati presentati da Kohn è riscrivere, o meglio, limitare la mappa affinchè la sua interpretazione combinatoria coincida con quella esplicita della mappa originale. Seguendo le regole esposte nei paragrafi precedenti, è necessario per prima cosa eliminare possibili ambiguità. L'unico caso presente riguarda la conversione stechiometrica del complesso $E2F : protease$ in $protease$, che non deve avvenire se la proteina $E2F$ è legata ad Rb. Per questo motivo bisognerebbe aggiungere un simbolo di inibizione che parta dal nodo $Rb : E2F$ e termini sulla freccia di conversione.

Per far combaciare l'interpretazione combinatoria e quella esplicita, bisogna fare in modo che i due legami non covalenti diventino mutuamente esclusivi. Si ottiene questo risultato aggiungendo due inibizioni con direzioni opposte. Queste contingenze rendono ridondante l'inibizione introdotta poco fà per bloccare la conversione, che può venir tranquillamente eliminata.

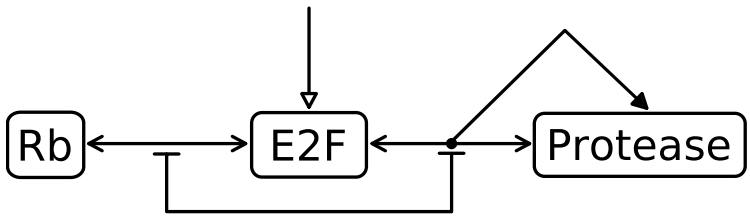


Figura 5.3: MIM di Figura 5.1 dopo aver applicato le regole di riscrittura per eliminare le ambiguità.

Le pochissime specie molecolari e reazioni in gioco non creano altre ambiguità per cui la versione finale della mappa è quella di Figura 5.3. Lo sforzo per eliminare le ambiguità e far coincidere le due interpretazioni si traduce solamente nell'aggiunta di due contingenze.

5.3 Il codice sCCP per la simulazione

5.3.1 Definizione delle specie molecolari

Grazie alle scelte effettuate nei precedenti paragrafi, per organizzare una simulazione di una rete così basta eseguire pochi semplici passi.
La prima cosa da fare è definire la descrizione della mappa, in termini di `molecular_type`, `port_id` e `contingencies`. Si parte quindi assegnando dei nomi ai nodi ed alle porte coinvolte, ottenendo come risultato la mappa di Figura 5.4. I nomi scelti per porte e nodi sono puramente arbitrari: *Rb* ed

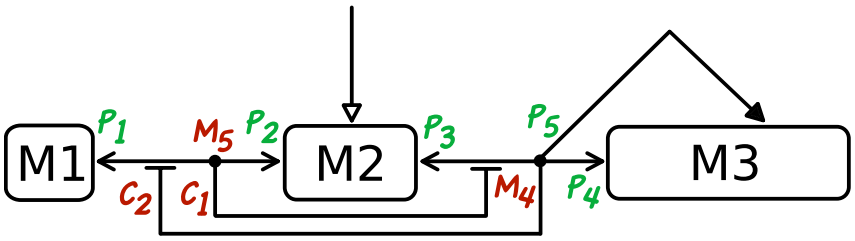


Figura 5.4: MIM simulata. Sono indicati i nomi delle porte, delle specie molecolari e delle contingenze.

E2F sarebbero andati bene ma per chiarezza si è deciso di uniformarli a dei nomi generici tipo M_1, M_2, ecc. Seguendo quindi i nomi della figura, le tre specie molecolari elementari si chiamano M_1, M_2 ed M_3, rispettivamente con gli insiemi di porte $[P_1]$, $[P_2, P_3]$ e $[P_4]$.

È necessario definire anche due nodi prodotti da interazioni: il primo è M_4, il risultato della complessazione tra M_2 ed M_3 (quindi tra *E2F* e *protease*), il quale ha la porta P_5 ed attiva la contingenza C_1. Il secondo è M_5, prodotto dalla complessazione tra M_1 ed M_2 (quindi tra *Rb* ed *E2F*) il quale attiva solamente la contingenza C_2.

Listato 5.1: I predicati che definiscono i nodi della mappa di Figura 5.4

```
molecular_type(m1, [p1], []).
molecular_type(m2, [p2, p3], []).
molecular_type(m3, [p4], []).
molecular_type(m4, [p5], [c1]).
molecular_type(m5, [], [c2]).
```

Nel Listato 5.1 sono riportati i 5 predicati che definiscono la struttura dei nodi appena descritti, secondo la sintassi definita in Tabella 4.1.

La definizione delle contingenze segue la sintassi presentata in Tabella 4.2: dopo il nome si dichiara se è una inibizione (specificando il valore 1) o una necessità (specificando 0), la lista di archi che la attiva ed infine la doppia lista di porte ed archi obiettivo di questa contingenza. Il Listato 5.2 riporta il predicato `contingencies` creato per questa mappa.

Listato 5.2: Il predicato delle contingenze della mappa di Figura 5.4

```
contingencies([
  [c1, 1, [[p3, p4]], [[p2], []]],
  [c2, 1, [[p1, p2]], [[p3], []]]
]).
```

Infine si definiscono le quantità iniziali delle molecole in gioco tramite il predicato `elementary_species`, illustrato nel Listato 5.3.

Listato 5.3: Il predicato che definisce le quantità iniziali per la mappa di Figura 5.4

```
elementary_species([
  [m1, 1000],
  [m2, 0],
  [m3, 1000],
  [m4, −1],
  [m5, −1]
  ]).
```

Per le quantità pari o superiori a 0 il simulatore creerà automaticamente i predicati `complex`, assegnando ad ognuno di loro un diverso `complex_id`. Le quantità impostate a -1 invece, servono a indicare i nodi che non sono specie molecolari elementari ma vengono solamente uniti ai complessi come conseguenza di una interazione.

5.3.2 Definizione delle reazioni

Per definire le reazioni è sufficiente instanziare gli agenti definiti in precedenza, specificando le porte che devono venir legate ed i relativi rate con i quali devono funzionare.

I primi due agenti a venir lanciati devono essere `port_man_spawner` ed il simmetrico per gli archi `edge_man_spawner` (si veda il Paragrafo 4.2.1), che si occuperanno di lanciare i gestori di ogni porta ed arco appena specificati. Quindi si può passare alla definizione delle reazioni vere e proprie: la mappa di Figura 5.4 specifica la sintetizzazione della specie molecolare M_3 ad una velocità costante, una conversione stechiometrica del complesso $M_2 : M_3$ in M_3 e le due complessazioni tra M_1 ed M_2 e tra M_2 ed M_3. La lista completa degli agenti istanziati per la simulazione è quella del Listato 5.4.

Listato 5.4: Gli agenti necessari per simulare la mappa di Figura 5.3

```
1 port_man_spawner@rate(fast, [])
2 & edge_man_spawner@rate(fast, [])
3 & production([m2,0], 8000)@rate(fast, [])
4 & conversion(p5, [[[[ m3 ,0]],[]]], 16)@rate(fast, [])
5 & complexation_mm(p1, p2, m5, 0.0001)@rate(fast, [])
6 & complexation_mm(p3, p4, m4, 0.004)@rate(fast, []).
```

Si noti che nella mappa originale di Kohn non sono previste le rotture dei legami non covalenti. Per questo motivo non sono stati instanziati agenti di decomplessazione, semplificando ulteriormente il codice sCCP.

5.4 Risultati della simulazione

Per eseguire la simulazione, a causa del doppio passo di interpretazione, è necessario lanciare l'interprete Prolog e fargli caricare l'interprete sCCP, il quale a sua volta eseguirà il codice appena discusso.

Durante la simulazione il motore di interpretazione fornisce come output le quantità dei `complex_type` che si sono formati dinamicamente. In questo modo è possibile tracciare la formazione di nuovi complessi e la loro diffusione nel sistema simulato.

Ad ulteriore conferma del fatto che le differenze tra le due interpretazioni

non si limitano alla semantica da attribuire alla mappa ma anche al comportamento dinamico del sistema si è provato a simulare la mappa di Figura 5.5 usando l'interpretazione combinatoria.

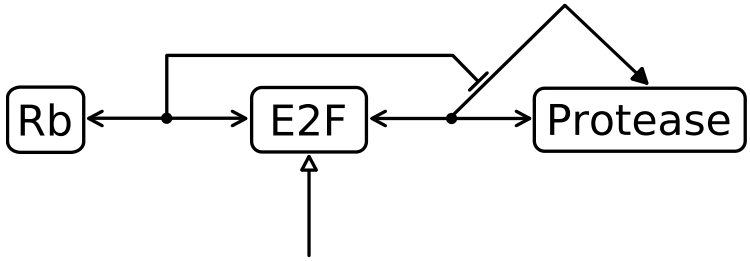


Figura 5.5: MIM interpretata usando l'interpretazione combinatoria.

L'unica differenza con la mappa presentata in precedenza sta nella mancanza della doppia inibizione tra le due reazioni di complessazione. Al loro posto è stata creata l'inibizione della conversione stechiometrica, seguendo le regole per eliminare le contingenze. L'andamento temporale del sistema è visualizzabile in Figura 5.9.

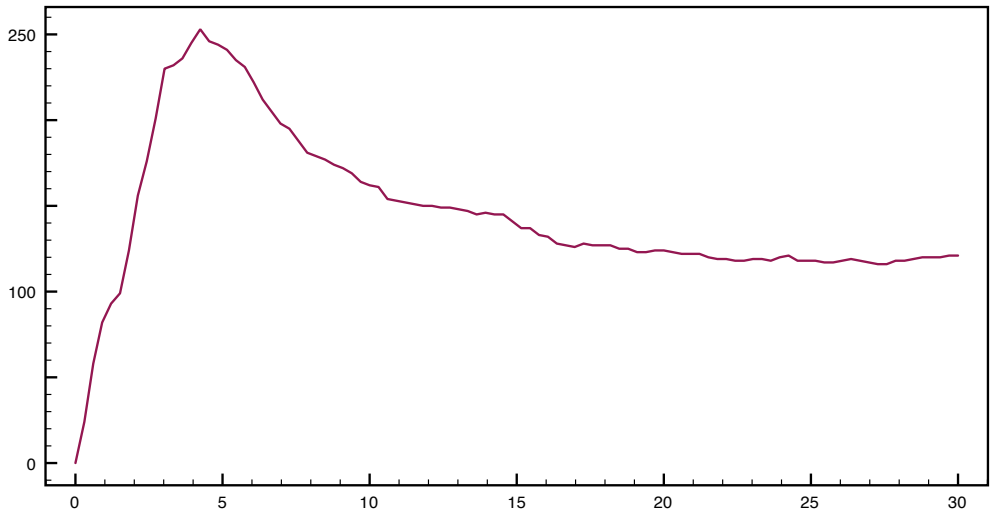


Figura 5.6: Andamento temporale del composto $Rb : E2F$ nel sistema descritto dalla mappa di Figura 5.5.

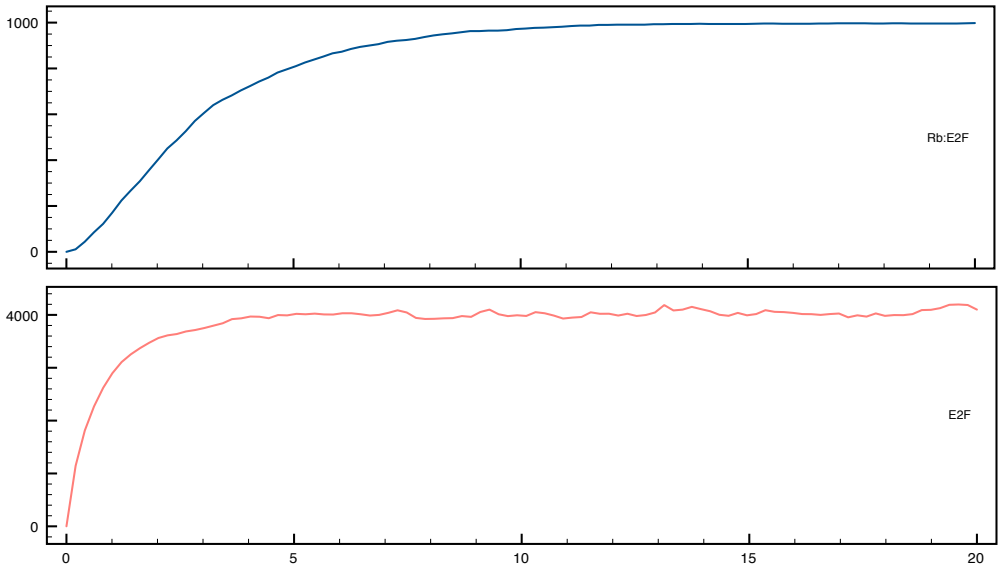


Figura 5.7: Andamento temporale della specie molecolare $E2F$ e del composto $Rb : E2F$ nel sistema descritto dalla mappa di Figura 5.4.

In Figura 5.7 invece è riportato l'andamento della simulazione della mappa di Figura 5.4, con la doppia inibizione. Si noti come il comportamento dinamico dei due sistemi sia diverso: nel primo parte dei complessi $E2F : protease$ si lega anche ad Rb, impedendo la loro degradazione tramite la conversione stechiometrica. Il secondo invece ha un andamento più semplice: una rapida crescita nella fase iniziale per poi rallentare fino ad assestarsi all'equilibrio del sistema.

5.5 Un secondo esempio

Seguendo l'articolo di Kohn ([19]), la mappa che descrive la transizione di fase viene raffinata, aggiungendo un fattore cdk che funge da chinasi per $Rb : E2F$: dopo essersi legato causa la dissociazione del complesso. La rete è quella di Figura 5.8, leggermente più complicata della precedente.

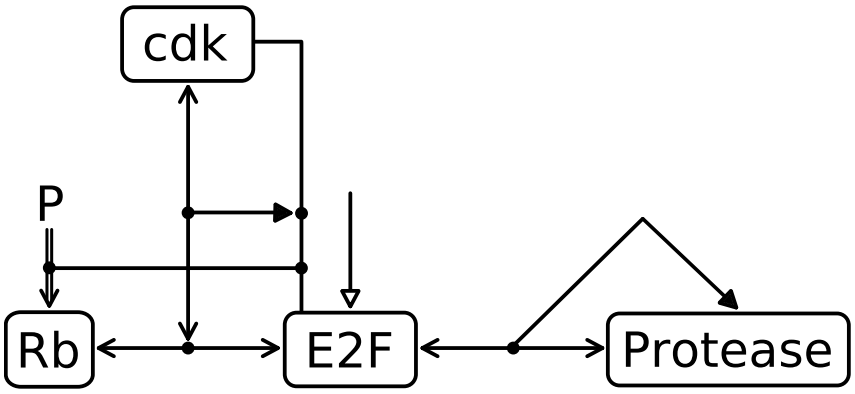


Figura 5.8: Evoluzione della rete di Figura 5.1

5.5.1 Definizione della mappa da simulare

Anche in questo caso l'applicazione delle regole di riscrittura per eliminare le ambiguità risulta semplice. Oltre alla doppia inibizione tra le due complessazioni di $E2F$, presente anche nella mappa precedente, bisogna preoccuparsi della fosforilazione di Rb, impedendo che essa avvenga quando è legato ad $E2F$ e viceversa. Questa doppia inibizione inoltre costringe le due interpretazioni combinatoria ed esplicita a coincidere, permettendo quindi ad una simulazione stocastica tramite sCCP di ottenere dei risultati confrontabili a quelli raggiunti nell'esperimento originale.

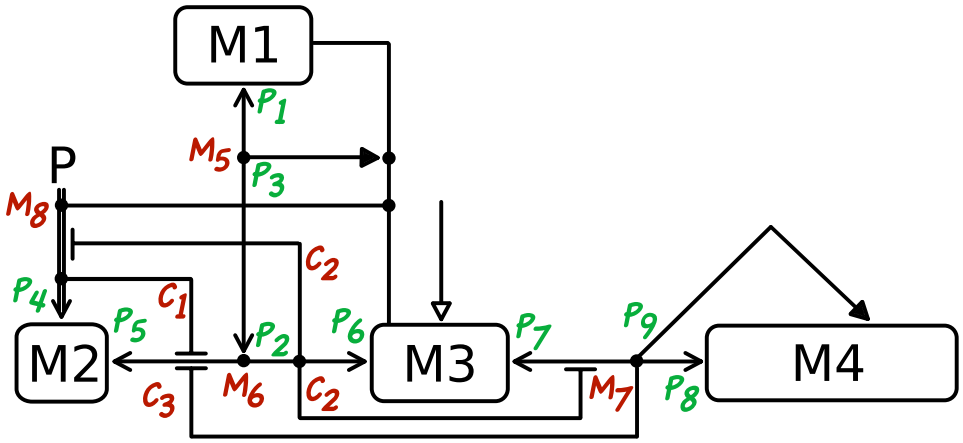


Figura 5.9: MIM simulata. Sono indicati i nomi delle porte, delle specie molecolari e delle contingenze.

Dopo aver assegnato dei nomi alle specie molecolari, alle porte ed alle contingenze, il risultato è la mappa di Figura 5.9. A questo punto è possibile definire il programma sCCP, partendo dalla definizione dei nodi come mostrato nel Listato 5.5.

Listato 5.5: I predicati che definiscono i nodi della mappa di Figura 5.9

```
1 molecular_type(m1, [p1], []) .
2 molecular_type(m2, [p4, p5], []) .
3 molecular_type(m3, [p6, p7], []) .
4 molecular_type(m4, [p8], []) .
5 molecular_type(m5, [p3], []) .
6 molecular_type(m6, [p2], [c2]) .
7 molecular_type(m7, [p9], [c3]) .
8 molecular_type(m8, [], [c1]) .
```

Le contingenze come detto sono quattro, due coppie di inibizioni reciproche, che si traducono in tre liste all'interno del predicato `contingency`. Questo accade perchè in verità due contingenze partono dalla stessa specie molecolare $M6$, permettendo quindi di sintetizzarle in C_2, sfruttando la sintassi della codifica. Le tre definizioni delle contingenze sono quelle del Listato 5.6.

Listato 5.6: Il predicato delle contingenze della mappa di Figura 5.9

```
1 contingencies([
2   [c1, 1, [[pp, p4]], [[p5], []]],
3   [c2, 1, [[p5, p6]], [[p4, p7], []]],
4   [c3, 1, [[p7, p8]], [[p6], []]]
5 ]) .
```

Avendo definito la topologia della mappa si procede con la definizione delle quantità iniziali, come visualizzato nel Listato 5.7.

Listato 5.7: Il predicato che definisce le quantità iniziali per la mappa di Figura 5.9

```
elementary_species([
  [m1, 20000],
  [m2, 60000],
  [m3, 0],
  [m4, 1000],
  [m5, −1],
  [m6, −1],
  [m7, −1],
  [m8, −1]
]).
```

La definizione delle interazioni è leggermente più complicata. Il punto centrale è la definizione dei prodotti della conversione stechiometrica originata dalla porta P_3 appartenente al nodo M_5: il primo è la specie molecolare elementare M_1; il secondo è la specie molecolare non elementare M_2 fosforilata, quindi un grafo di due nodi ed un arco; il terzo ed ultimo la specie molecolare elementare M_3. Per concludere, rispetto all'esempio precedente, è sufficiente aggiungere una reazione di complessazione tra *cdk* ed $Rb : E2F$, quindi tra le porte P_1 e P_2. La lista degli agenti che definiscono le interazioni della mappa è quella del Listato 5.8.

Listato 5.8: Gli agenti necessari per simulare la mappa di Figura 5.9

```
port_man_spawner@rate(fast, [])
& edge_man_spawner@rate(fast, [])
& production([m3,0], 8000)@rate(fast, [])
& conversion(p3, [[[[ m1 ,0]],[]], [[[ m3 ,0]],[]], [[[[ m2 ,0],[ m8 ,1]]], [[[0, p4
    ],[1, pp ]]]]], 16)@rate(fast, []) @rate(fast, [])
& conversion(p9, [[[[ m4 ,0]],[]]]) @rate(fast, [])
& complexation_mm(p5, p6, m6, 0.5)@rate(fast, [])
& complexation_mm(p7, p8, m7, 0.005)@rate(fast, [])
& complexation_mm(p1, p2, m5, 0.005)@rate(fast, []).
```

5.5.2 Risultati della simulazione

I risultati ottenuti dalla simulazione della mappa di Figura 5.9 sono visualizzati in Figura 5.10.

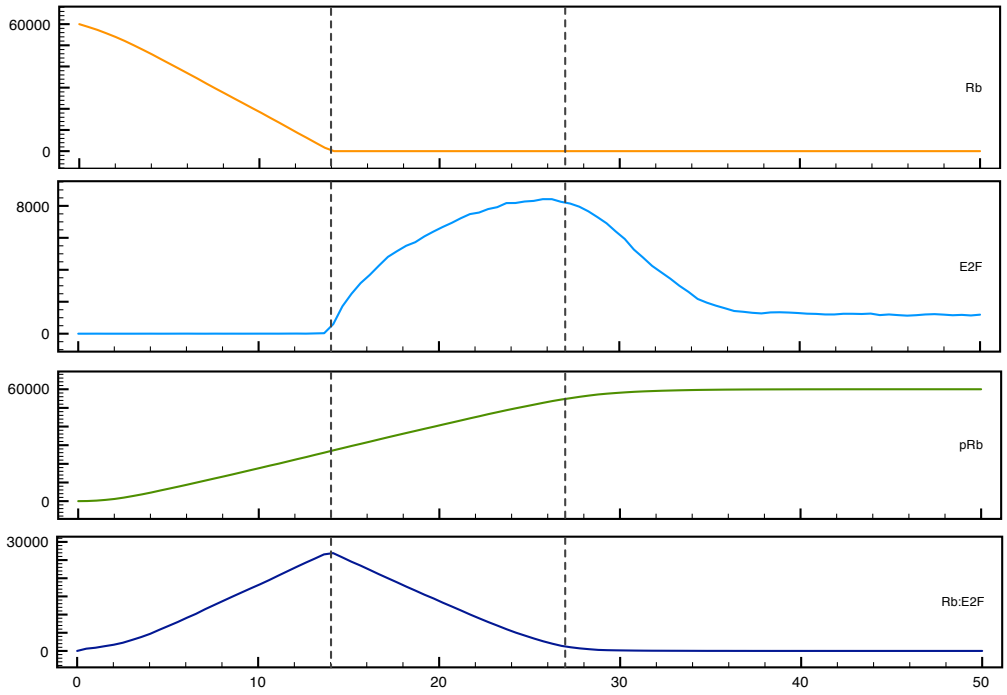


Figura 5.10: Andamenti temporali dei vari composti simulati dalla rete di Figura 5.9

Per mantenere il comportamento della mappa semplice, la fosforilazione non è inclusa tra le reazioni: il complesso pRb appare solo come prodotto della conversione originata da $Rb : E2F : cdk$. La prima fase della simulazione, fino alla prima linea tratteggiata verticale di Figura 5.10, evidenzia come gli $E2F$ si leghino velocemente agli Rb mentre pRb salga con andamento praticamente lineare. Durante questa fase c'è un temporaneo accumulo di $Rb : E2F$.

Una volta terminati gli Rb liberi, tra la prima e la seconda riga tratteggiata verticale, inizia lo smaltimento degli $Rb : E2F$ ad opera della conversione tramite cdk, mantenendo lineare l'andamento di crescita di pRb e determinando la diminuzione dei complessi $Rb : E2F$. Gli pRb non sono più disponibili per essere legati ad $E2F$, i quali iniziano rapidamente ad aumentare.

Nella terza fase, dopo la seconda linea tratteggiata verticale, gli Rb sono presenti ormai esclusivamente in forma fosforilata, i complessi $Rb : E2F$ sono stati interamente smaltiti e la quantità di $E2F$ inizia a calare, fino ad assestarsi su un equilibrio tra produzione e degradazione.

Il modello appena simulato descrive come dei fattori tipo *Rb* ed *cdk* siano in grado di generare un'andamento ad onda di $E2F$. Volendo fare un parallelo con i circuiti elettrici, si potrebbe dire che *Rb* funge da capacità, immagazzinando gli $E2F$ per poi rilasciarli gradualmente.

Le simulazioni presentate dimostrano quanto il presente approccio sia flessibile e si possa adattare a qualsiasi situazione. Si può facilmente modificare una mappa limitandone i comportamenti e vedere quali effetti si propagano in fase di esecuzione della stessa.

Capitolo 6

Considerazioni finali

6.1 Lavori collegati

Si passano in rassegna ora, in modo sintetico, alcuni tra i più significativi approcci simili a quello presentato.

Primo fra tutti è sicuramente quello adottato da Kohn stesso ([19], [21]) che, come detto, utilizza l'interpretazione esplicita per interpretare le MIM che disegna per poi generare un sistema di Equazioni Differenziali da risolvere seguendo il principio di mass-action. Tuttavia ricavare l'intera lista di reazioni può essere un compito non banale a causa della dimensione della mappa, come discusso precedentemente.

Un altro approccio interessante è quello del *k-calcolo* [8], una algebra di processo supportata da una semantica operazionale sotto forma di regole di riscrittura, basata sulle operazioni primitive di complessazione e attivazione (una generalizzazione della fosforilazione) per cui di recente sono stati proposti dei metodi stocastici per simularne i costrutti [9].

Una estensione del noto π-*calcolo* [6], chiamato β-*binders* ([7], [24]), nasconde i π-processi in delle scatole che possono interagire tra loro tramite dei punti di interazione esposti all'esterno, appunto i *binders*. Una primitiva di comunicazione permette a diverse scatole di interagire su dei canali tipati. Queste scatole inoltre possono venir divise o unite a tempo di esecuzione, cosa che le rende decisamente flessibili e potenti. Le MIM possono venir descritte in termini di β-binders assumendo per ipotesi che le molecole siano rappresentate come scatole e caratterizzate in modo completo dai loro siti di interazione, modellati come binders. All'interno di ogni scatola ci sono dei π-processi che

definiscono tutte le interazioni possibili tramite i proprio binders. Purtroppo per poter aggiungere un nuovo tipo di interazione è necessario modificare tutti i π-processi coinvolti. Nella codifica presentata da questa tesi, invece, si associa un agente ad ogni freccia di reazione presente sulla mappa, quindi aggiungere una reazione si traduce in una semplice aggiunta di un agente in parallelo agli altri. In più i punti di interazione sono gestiti da agenti specifici che vengono lanciati automaticamente dopo aver letto la descrizione della mappa. Anche l'aggiunta di nuovi tipi molecolari dunque risulta semplice e veloce e non richiede la modifica di quanto specificato in precedenza.

Un'altra differenza sostanziale sta nel fatto che l'approccio seguito in questa tesi non crea un nuovo linguaggio, ma semplicemente ne utilizza uno già esistente sfruttandone quelle caratteristiche che ben si adattano ai sistemi che si intende descrivere. sCCP inoltre offre delle possibilità che non sono state indagate ma solo identificate, come la possibilità di utilizzare un tipo di cinetica chimica diversa dalla mass-action (come per esempio quella di Michaelis-Menten), semplicemente tramite la definizione dei rate.

6.2 Conclusioni

In questo lavoro di tesi si è presentata una implementazione in Stochastic Concurrent Constraint Programming (sCCP) di un algoritmo per la simulazione di entità e processi biologici definite tramite le Mappe di Interazione Molecolare (MIM). Si è seguita l'interpretazione combinatoria che permette di mantenere bassa la complessità della mappa pur descrivendo un gran numero di interazioni. Questo è ottenuto grazie a dei comportamenti impliciti: ogni arco in una MIM infatti rappresenta una famiglia di reazioni potenzialmente molto grande.

La codifica proposta è capace di simulare queste mappe senza dover generare l'intera lista di reazioni, mantendendola implicita grazie ad una rappresentazione che genera solo le entità che effettivamente vengono generate a tempo di esecuzione. Essendo basato sui grafi, risultano molto semplici le operazioni di unione e divisione di complessi, inoltre l'idea di specificare le reazioni in termini delle porte coinvolte, permette di non dover elencare l'intero insieme delle reazioni espresse dalla mappa.
È stato scritto un prototipo di simulatore in SICStus Prolog, ed utilizzato per alcuni semplici test per dimostrare l'efficacia dell'approccio.

La scelta di sCCP come linguaggio per rappresentare simili costrutti è giustificata principalmente dalla sua componente stocastica, perfetta per modellare

entità biologiche. L'evoluzione autonoma del sistema è stata delegata interamente alla semantica stocastica del linguaggio. Oltre a ciò, la potenza del constraint store e dei vincoli esprimibili in Prolog ha permesso di descrivere i grafi associati ai complessi in maniera da poterli utilizzare direttamente per definire e tracciare l'evoluzione del sistema. Non bisogna sottovalutare la composizionalità rispetto all'aggiunta di archi e nodi che il presente approccio mantiene, come descritto nei paragrafi precedenti.

Inoltre, la lunghezza della descrizione di una mappa in sCCP risulta proporzionale al numero di simboli che essa contiene, quindi evita possibili esplosioni di complessità causate dalle traduzioni ma soprattutto non soffre la complessità intrinseca dell'interpretazione combinatoria.

Come detto il simulatore è, per il momento, poco più di un prototipo. Il doppio livello di interpretazione richiesto per effettuare una simulazione non assicura di certo prestazioni eccezionali. Sicuramente uno dei primi miglioramenti sarà quello di riscrivere il metainterprete sCCP, magari includendolo in una specie di sottoinsieme di linguaggio logico tipo Prolog, snellito di tutte le librerie e funzionalità inutili dal punto di vista di sCCP stesso. In questo modo si eviterebbe il doppio passo di interpretazione, decisamente poco performante.

L'interprete per le mappe potrebbe venir equipaggiato di una interfaccia grafica grazie alla quale poter disegnare le mappe. Tale interfaccia potrebbe generare automaticamente tutti i particolari delle descrizioni di specie molecolari, reazioni, porte e contingenze di cui il simulatore ha bisogno. Uno strumento di questo tipo potrebbe rivelarsi un efficace ponte tra il simulatore ed il mondo dei biologi, notoriamente poco avvezzi ad interagire con linguaggi di programmazione come Prolog, permettendo una sua ampia diffusione. Simmetricamente è possibile definire una interfaccia per la raccolta dei risultati o per la loro visualizzazione in tempo reale, altro ingrediente che sicuramente farebbe piacere a degli effettivi utilizzatori.

Dal punto di vista della codifica invece, dei miglioramenti potrebbero riguardare il problema dell'isomorfismo dei grafi. Soprattutto in presenza di reazioni di omodimerizzazione infatti, risulta decisamente difficile verificare se due grafi associati ad altrettanti complessi siano a tutti gli effetti uno solo.

L'approccio si è dimostrato efficace e funzionale già dai primi test, offrendo buoni risultati anche in fase di simulazione. Unitamente ai miglioramenti descritti si potrebbe rivelare un importante strumento per indagare entità e comportamenti biologici a livello molecolare.

Bibliografia

[1] M. Blinov, J. Yang, J. Faeder, and W. Hlavacek. *Graph theory for rule-based modeling of biochemical networks*, Transactions of Computational Systems Biology, 2007.

[2] R. Blossey, L. Cardelli, A. Phillips. *A Compositional Approach to the Stochastic Dynamics of Gene Networks*, 2006

[3] L. Bortolussi. *Stochastic concurrent constraint programming*, In Proceedings of 4th International Workshop on Quantitative Aspects of Programming Languages, QAPL 2006, ENTCS, volume 164, pages 6580, 2006.

[4] L. Bortolussi. *Constraint-based approaches to stochastic dynamics of biological systems*, PhD thesis, PhD in Computer Science, University of Udine, 2007.

[5] L. Bortolussi and A. Policriti. *Modeling biological systems in concurrent constraint programming*, In Proceedings of Second International Workshop on Constraint-based Methods in Bioinformatics, WCB 2006, 2006. press, 1993.

[6] L. Cardelli. *Chemical π-calculus*, www.lucacardelli.name, 2007.

[7] F. Ciocchetta, C. Priami, and P. Quaglia. *Modeling Kohn interaction maps with beta-binders: an example*, Transactions on computational systems biology, LNBI 3737:3348, 2005.

[8] V. Danos and C. Laneve. *Formal molecular biology*, Theor. Comput. Sci., 325(1):69-110, 2004.

[9] V. Danos, J. Feret, W. Fontana, R. Harmer, J. Krivine. *Rule-based modelling of cellular signalling*, CONCUR'07 invited paper, 2007.

[10] H. De Jong. *Modeling and simulation of genetic regulatory systems: A literature review*, Journal of Computational Bi- ology, 9(1):67103, 2002.

[11] L. Edelstein-Keshet. *Mathematical Models in Biology*, SIAM, 2005.

[12] D. Gillespie. *A general method for numerically simulating the stochastic time evolution of coupled chemical reactions*, J. of Computational Physics, 22, 1976.

[13] D. Gillespie. *Exact stochastic simulation of coupled chemical reactions*, J. of Physical Chemistry, 81(25), 1977.

[14] J. Jaffar and M. J. Maher. *Constraint logic programming: a survey*, Journal of logic programming, 1994.

[15] J. Jaffar, M. H. Maher, K. Marriott and P. H. Stuckey. *The semantics of constraint logic programs*, Journal of logic programmin, 37(1-2):1-46, 1998.

[16] H. Kitano. *Foundations of Systems Biology*, MIT Press, 2001.

[17] H. Kitano. *Computational systems biology*, Nature, 420:206210, 2002.

[18] K. W. Kohn, M. I. Aladjem, J. N. Weinstein, and Y. Pommier. *Molecular interaction maps of bioregulatory networks: A general rubric for systems biology*, Molecular Biology of the Cell, 17(1):113, 2006.

[19] K. W. Kohn. *Functional capabilities of molecular network components controlling the mammalian g1/s cell cycle phase transition*, Oncogene, 16:10651075, 1998.

[20] K. W. Kohn. *Molecular interaction map of the mammalian cell cycle control and dna repair systems*, Molecular Biology of the Cell, 10:27032734, August 1999.

[21] K. W. Kohn, M. I. Aladjem, S. Kim, J. N. Weinstein, and Y. Pommier. *Depicting combinatorial complexity with the molecular interaction map notation*, Molecular Systems Biology, 2(51), 2006.

[22] J. R. Norris. *Markov Chains*, Cambridge University Press, 1997.

[23] C. Priami and P. Quaglia. *Modelling the dynamics of biosystems*, Briefings in Bioinformatics, 5(3):259269, 2004.

[24] C. Priami and P. Quaglia. *Beta binders for biological interactions*, In Proceedings of Computational methods in system biology (CMSB04), 2005.

[25] C. Priami and A. Romanel. *The decidability of the structural congruence for beta-binders*, In MeCBIC 2006: Workshop on Membrane Computing and Biological ly Inspired Process Calculi, ENTCS. Elsevier, 2006.

[26] Romanel, A., Dematte, L., Priami, C. *The Beta Workbench* Available from http://www.cosbi.eu/Rpty_Tech.php

[27] V. A. Saraswat. *Concurrent Constraint Programming*, MIT press, 1993.

[28] V. A. Saraswat, M. Rinard, and P. Panangaden. *Semantics foundations of concurrent constraint programming*, in Proceedings of POPL, 1991.

[29] *http://www.sics.se/isl/sicstuswww/site/index.html*, SICStus Prolog home page.

[30] E. O. Voit. *Computational analysis of biochemical systems*, Cambridge University Press, 2000.

[31] D. J. Wilkinson. *Stochastic Modelling for Systems Biology*, Chapman and Hall, 2006.

Ringraziamenti

Il primissimo grazie va alla nonna Norma, che, lungimirante, prima di lasciarci ha sottolineato il suo amore piantando il seme per questa mia avventura Pisana. Grazie.

△▽△

Voglio ringraziare i miei genitori, che hanno innaffiato e curato sto seme in lungo e in largo, rendendo rigogliosa e spensierata la mia permanenza, che mi sono sempre stati vicini in barba ai chilometri, che mi hanno sempre permesso di pensare, dire e fare quello che volevo. Se sono arrivato fin qui è soprattutto merito loro.

△▽△

Grazie al mio fratellone preferito Jacopo per volermi così bene e per dimostrarmelo a modo suo con i continui confronti, i generosi consigli, le visite, i capolavori musicali che crea e mi spedisce, le partite a quel *giochino di melma* e tanto, tanto altro.

△▽△

Grazie a Danae, che se lo meriterebbe soltanto per il fatto di avermi sopportato in questi furiosi ultimi giorni. Inutile elencare le azioni materiali in cui giornalmente si prodiga per me, sarebbero davvero troppe. Spero continui ad essere energica, vitale oltre che insaziabilmente curiosa: a me piace così. Nonostante i suoi sforzi, il kebab migliore rimane quello alla stazione.

Un grazie particolare agli amici che mi hanno fatto sentire il loro affetto divorando più volte tonnellate d'autostrada, con il bello e col cattivo tempo, d'estate e d'inverno. Senza un *come xè cio* a spezzare il ritmo dei *boia de*, non ce l'avrei mai potuta fare. Grazie per i momenti a Trieste e dintorni, le cene, le serate ma soprattutto per *non eserse mai lasai come cani*.

△▽△

Grazie anche a tutti gli amici conosciuti in loco, Pisani della città, Pisani della provincia ed anche ai non Pisani. Le mille sagre, le gite, le cene, i pranzi, le merende, le partite virtuali e non... insomma... grazie a tutti di tutto!

△▽△

Grazie ai quei mattachioni della capoeira, per le ore di sudato divertimento (l'aggettivo non è scelto a caso) e per gli sforzi nel portare avanti il progetto nonostante le avversità. Mi raccomando non interrompete il cicitontin!

△▽△

Grazie a chi ha condiviso con me molto più di un tetto sotto il quale dormire, quei 4 che mi hanno dimostrato in molti modi la loro amicizia (chi più chi meno) e hanno assecondato le mie fissazioni e perdonato gli scatti d'ira. A questa divertente combriccola allego il padrone di casa, più unico che raro. Ad ognuno di loro: grazie Gino!

△▽△

Infine l'immancabile, tradizionale e sempreverde:
saluto tutti quelli che mi conoscono!